

---

# **methylcheck Documentation**

***Release 0.8.4***

**FOXO Technologies, inc**

**Feb 08, 2022**



---

## Contents:

---

<b>1</b>	<b>methylcheck is part of the methylsuite</b>	<b>3</b>
<b>2</b>	<b>Methylsuite package components</b>	<b>5</b>
<b>3</b>	<b>Installation</b>	<b>7</b>
<b>4</b>	<b>Tutorials and Guides</b>	<b>9</b>
4.1	Loading processed methylation data and checking beta distributions . . . . .	9
4.1.1	Loading Beta Values and Metadata . . . . .	9
4.1.2	Loading Other Types of Data . . . . .	11
4.1.3	Loading Data from .csv Files . . . . .	15
4.1.4	Checking Beta Distributions . . . . .	17
4.2	Filtering Poor Quality Probes . . . . .	18
4.2.1	Available probe exclusion lists . . . . .	19
4.2.2	Filtering poor quality probes . . . . .	19
4.2.3	Filtering sex-linked probes and control probes . . . . .	22
4.3	Quality Control . . . . .	25
4.3.1	controls_report (a color-coded spreadsheet of control probe performance per sample)	25
4.3.2	Quality Control in the IDE . . . . .	28
4.3.3	Predicting Sex . . . . .	42
4.3.4	Report PDF tool . . . . .	43
4.4	Custom QC with pOOBAH Vales . . . . .	45
4.4.1	Load the Beta Values in a dataframe . . . . .	45
4.4.2	Load p-values in a dataframe . . . . .	48
4.4.3	Mask Beta values where probe fails . . . . .	49
4.4.4	Remove Samples based on Percent or Number of Failed Probes . . . . .	51
4.4.5	Drop out Probes with a Percentage of NaNs . . . . .	52
4.5	Outlier detection using Multidimensional Scaling (MDS) . . . . .	54
4.6	API Reference . . . . .	72
4.6.1	methylcheck.cli . . . . .	73
4.6.2	methylcheck.run_pipeline . . . . .	74
4.6.3	methylcheck.run_qc . . . . .	74
4.6.4	methylcheck.read_geo . . . . .	75
4.6.5	methylcheck.load . . . . .	76
4.6.6	methylcheck.load_both . . . . .	77
4.6.7	methylcheck.qc_signal_intensity . . . . .	78
4.6.8	methylcheck.plot_M_vs_U . . . . .	79

4.6.9	methylcheck.plot_controls	79
4.6.10	methylcheck.plot_beta_by_type	80
4.6.11	methylcheck.probes	80
4.6.12	methylcheck.list_problem_probes	80
4.6.13	methylcheck.exclude_probes	81
4.6.14	methylcheck.exclude_sex_control_probes	82
4.6.15	methylcheck.drop_nan_probes	83
4.6.16	methylcheck.samples	83
4.6.17	methylcheck.sample_plot	83
4.6.18	methylcheck.beta_density_plot	83
4.6.19	methylcheck.mean_beta_plot	84
4.6.20	methylcheck.mean_beta_compare	84
4.6.21	methylcheck.beta_mds_plot	84
4.6.22	methylcheck.combine_mds	85
4.6.23	methylcheck.cumulative_sum_beta_distribution	86
4.6.24	methylcheck.predict	86
4.6.25	methylcheck.get_sex	86
4.6.26	methylcheck.assign	87
4.6.27	Loading Data	87
4.6.28	ReportPDF Report Builder class	91
4.6.29	Run QC pipeline	93
4.6.30	filtering probes	94
4.6.31	plotting functions	97
4.6.32	sex prediction	103
4.7	Release History	104
4.7.1	v0.8.3	104
4.7.2	v0.8.2	104
4.7.3	v0.8.1	104
4.7.4	v0.8.0	104
4.7.5	v0.7.9	105
4.7.6	v0.7.6	105
4.7.7	v0.7.5	105
4.7.8	v0.7.4	106
4.7.9	v0.7.3	106
4.7.10	v0.7.2	106
4.7.11	v0.7.1	106
4.7.12	v0.7.0	107
4.7.13	v0.6.4	107
4.7.14	v0.6.3	107
4.7.15	v0.6.2	107
4.7.16	v0.6.1	107
4.7.17	v0.6.0	107
4.7.18	v0.5.9	108
4.7.19	v0.5.7	108
4.7.20	v0.5.4	108
4.7.21	v0.5.2	108
4.7.22	v0.5.1	108
4.7.23	v0.5.0	108
4.7.24	v0.4.0	108
<b>5</b>	<b>Indices and tables</b>	<b>111</b>
	<b>Python Module Index</b>	<b>113</b>





`methylcheck` is a Python-based package for filtering and visualizing Illumina methylation array data. The focus is on quality control. View on [ReadTheDocs](#).

```
[3]: 6285625091_R05C01  7796806148_R03C02  7796806148_R01C02  6285625091_R03C02
```

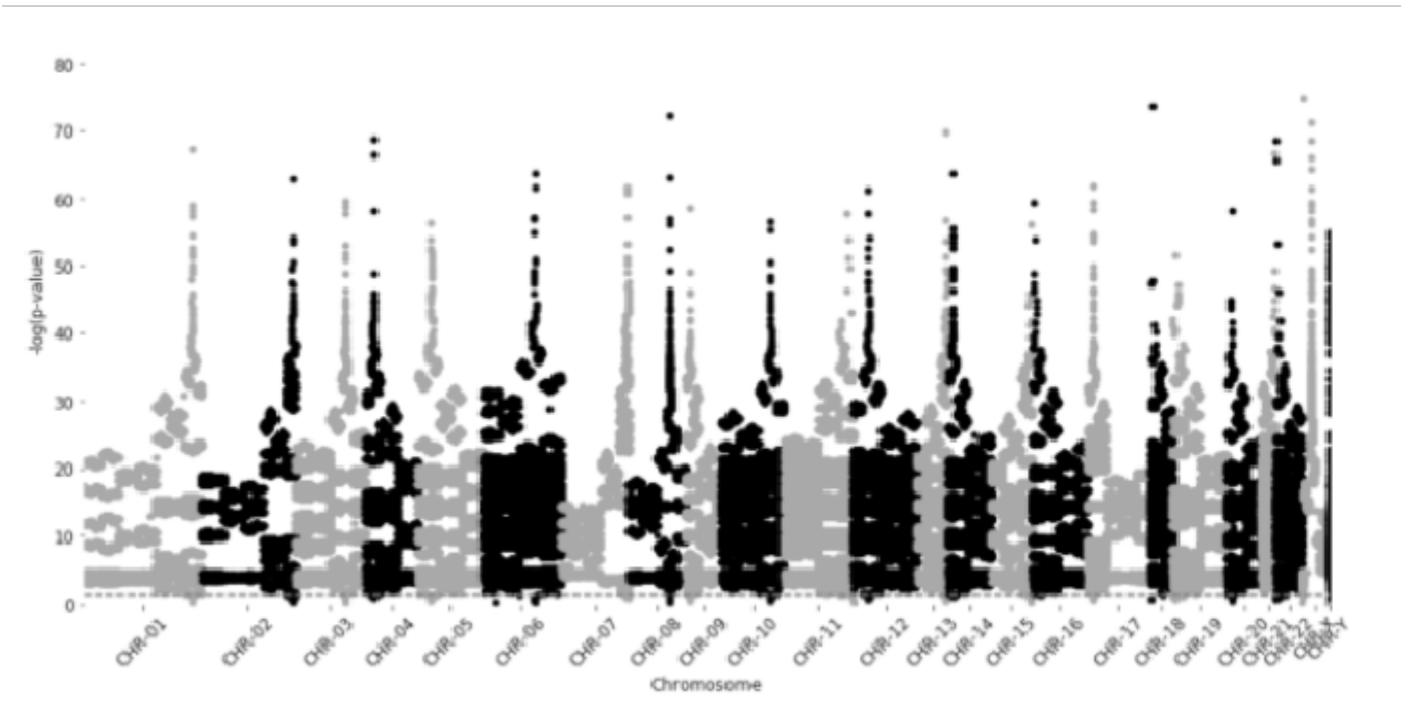
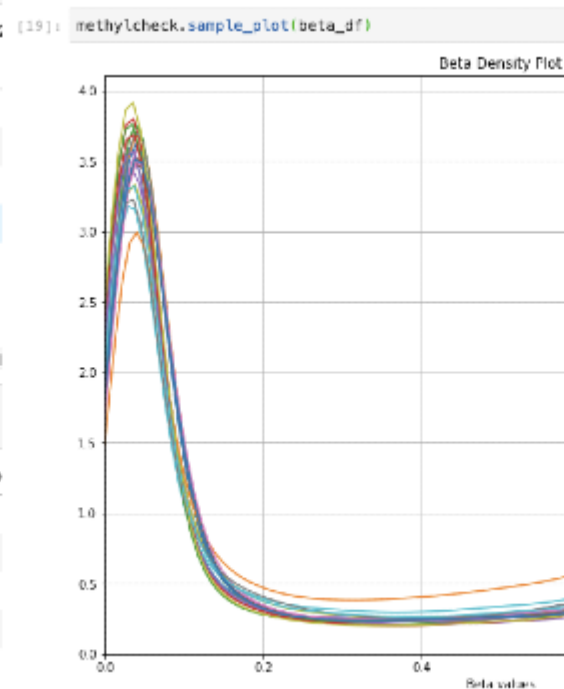
ilmnID	6285625091_R05C01	7796806148_R03C02	7796806148_R01C02	6285625091_R03C02
cg00035864	-0.514961	-2.015454	-1.958699	
cg00061679	-0.647032	0.616919	0.547593	
cg00063477	-0.639987	4.207996	3.884379	
cg00121626	-0.876430	-0.253725	-0.271588	
cg00223952	-4.638848	-4.387776	-4.630668	

5 rows x 21 columns

```
[4]: meta_df = pd.read_pickle('sample_sheet_meta_data.pkl')
      meta_df.head()
```

```
[4]:
```

	CheeZ	BuffyCoat	Sentrix_ID	Sentrix_Position	Sample_Group	Sample_Name	Samp
0	1	0	6285625091	R05C01	None	8.24 CD34	
1	1	0	7796806148	R03C02	None	7.25 PMN	
2	1	0	7796806148	R01C02	None	7.25 PROS	
3	1	0	6285625091	R06C02	None	9.1 PMN	
4	2	0	6285625091	R03C01	None	8.10 CD19	





# CHAPTER 1

---

## methylcheck is part of the methylsuite

---

`methylcheck` is part of the `methylsuite` of python packages that provide functions to process and analyze DNA methylation data from Illumina's Infinium arrays (27k, 450k, and EPIC, as well as mouse arrays). This package is focused on quality control for processed methylation data.

`methylcheck` 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.



---

### 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 of staining, bisulfite conversion, hybridization, extension, negative, non-polymorphic, target removal, and specificity
  - spreadsheet summary of control probe performance
  - 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)

- differentially methylated regions, with gene annotation from the UCSC Human Genome Browser
- volcano plots (to identify probes that are the most different)
- manhattan plots (to identify clusters of probes associated with genomic regions that are different)

## CHAPTER 3

---

### Installation

---

`methylcheck` maintains configuration files for your Python package manager of choice: `pipenv` or `pip`. Conda install is coming soon.

```
>>> pip install methylcheck
```

or if you want to install all three packages at once (recommended):

```
>>> pip install methylsuite
```



If you are new to DNA methylation analysis, we recommend reading through this [introduction](#) from the methylprep documentation. Otherwise, you are ready to use methylcheck to:

- *load processed methylation data*
- *filter unreliable probes from your data*
- *run array-level quality control reports*
- *detect outlier samples*
- *predict the sex of human samples*

## 4.1 Loading processed methylation data and checking beta distributions

### 4.1.1 Loading Beta Values and Metadata

We will continue working with the dataset referenced in the methylprep general walkthrough: [GSE147391](#). It was processed from the command line with the following command:

```
>>> python -m methylprep process -d <filepath> --all
```

Where <filepath> is the directory where the raw IDAT files are stored. We use `--all` to tell methylprep to give us all the possible output files, which include:

- `beta_values.pkl`
- `poobah_values.pkl`
- `control_probes.pkl`
- `m_values.pkl`
- `noob_meth_values.pkl`

- noob\_unmeth\_values.pkl
- meth\_values.pkl
- unmeth\_values.pkl
- sample\_sheet\_meta\_data.pkl

As well as folders that contain the processed methylation data in .csv files.

```
[1]: import methylcheck
      from pathlib import Path
      filepath = Path('/Users/patriciagirardi/tutorial/GPL21145')

      betas = methylcheck.load(filepath)
      betas.head()
```

```
Files: 100%| 1/1 [00:00<00:00, 5.92it/s]
INFO:methylcheck.load_processed:loaded data (865859, 16) from 1 pickled files (0.222s)
```

```
[1]:
```

	203163220027_R01C01	203163220027_R02C01	203163220027_R03C01	\
IlmnID				
cg000000029	0.852	0.749	0.739	
cg000000103	NaN	NaN	NaN	
cg000000109	0.943	0.916	0.884	
cg000000155	0.960	0.962	0.958	
cg000000158	0.969	0.972	0.971	
	203163220027_R04C01	203163220027_R05C01	203163220027_R06C01	\
IlmnID				
cg000000029	NaN	0.891	0.896	
cg000000103	NaN	NaN	NaN	
cg000000109	0.951	0.936	0.774	
cg000000155	0.958	0.963	0.959	
cg000000158	0.968	0.968	0.964	
	203163220027_R07C01	203163220027_R08C01	203175700025_R01C01	\
IlmnID				
cg000000029	0.757	0.734	0.806	
cg000000103	NaN	NaN	NaN	
cg000000109	0.932	0.920	0.898	
cg000000155	0.965	0.969	0.959	
cg000000158	0.970	0.974	0.964	
	203175700025_R02C01	203175700025_R03C01	203175700025_R04C01	\
IlmnID				
cg000000029	0.881	0.740	0.885	
cg000000103	NaN	NaN	NaN	
cg000000109	0.938	0.931	0.953	
cg000000155	0.956	0.964	0.961	
cg000000158	0.969	0.960	0.968	
	203175700025_R05C01	203175700025_R06C01	203175700025_R07C01	\
IlmnID				
cg000000029	0.693	0.779	0.617	
cg000000103	NaN	NaN	NaN	
cg000000109	0.899	0.889	0.734	
cg000000155	0.959	0.967	0.961	
cg000000158	0.965	0.965	0.975	

(continues on next page)



(continued from previous page)

```

203175700025_R08C01
IlmnID
cg00000029      0.891
cg00000103      NaN
cg00000109      0.892
cg00000155      0.960
cg00000158      0.966

```

You may also use `methylcheck.load_both()` to load both the beta values and metadata at the same time. Note that methylcheck expects the formatting used by methylprep in this command.

```

[2]: df, meta = methylcheck.load_both(filepath)
      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:['/Users/patriciagirardi/tutorial/
↳GPL21145/GSE147391_GPL21145_meta_data.pkl', '/Users/patriciagirardi/tutorial/
↳GPL21145/sample_sheet_meta_data.pkl']
INFO:methylcheck.load_processed:Multiple meta_data found. Only loading the first file.
INFO:methylcheck.load_processed:Loading 16 samples.
Files: 100%|| 1/1 [00:00<00:00, 5.98it/s]
INFO:methylcheck.load_processed:loaded data (865859, 16) from 1 pickled files (0.2s)
INFO:methylcheck.load_processed:meta.Sample_IDs match data.index (OK)

[2]:
      GSM_ID      Sample_Name      Sentrrix_ID Sentrrix_Position \
2   GSM4429898   Grade II rep3   203163220027      R01C01
3   GSM4429899   Grade III rep1   203163220027      R02C01
12  GSM4429908   Grade IV rep5   203163220027      R03C01
15  GSM4429911   Grade IV rep8   203163220027      R04C01
6   GSM4429902   Grade II rep6   203163220027      R05C01

      source      histological diagnosis description \
2   Resected glioma      Diffuse astrocytoma (II)      Glioma
3   Resected glioma      Anaplastic astrocytoma (III)      Glioma
12  Resected glioma      Glioblastoma (IV)      Glioma
15  Resected glioma      Glioblastoma (IV)      Glioma
6   Resected glioma      Oligodendroglioma (II)      Glioma

      Sample_ID
2   203163220027_R01C01
3   203163220027_R02C01
12  203163220027_R03C01
15  203163220027_R04C01
6   203163220027_R05C01

```

## 4.1.2 Loading Other Types of Data

Formats supported by `methylcheck.load()` are:

```
['beta_value', 'm_value', 'meth', 'meth_df', 'noob_df', 'sesame', 'beta_csv']
```

where 'beta\_value' is the default.

```
[3]: # the unnormalized probe values
(meth,unmeth) = methylcheck.load(filepath, format='meth_df')
meth.head()
```

```
100%| 16/16 [00:00<00:00, 488.67it/s]
100%| 16/16 [00:00<00:00, 1480.91it/s]
INFO:methylcheck.load_processed:(865859, 16) (865859, 16)
```

```
[3]:      203163220027_R01C01  203163220027_R02C01  203163220027_R03C01  \
IlmnID
cg000000029          933.0          874.0          845.0
cg000000103           NaN           NaN           NaN
cg000000109         2597.0         2428.0         2309.0
cg000000155         3491.0         4397.0         3956.0
cg000000158         5556.0         5534.0         5384.0

      203163220027_R04C01  203163220027_R05C01  203163220027_R06C01  \
IlmnID
cg000000029          528.0         1579.0         1243.0
cg000000103           NaN           NaN           NaN
cg000000109         3082.0         2542.0         2382.0
cg000000155         4059.0         4314.0         4059.0
cg000000158         4833.0         5381.0         6078.0

      203163220027_R07C01  203163220027_R08C01  203175700025_R01C01  \
IlmnID
cg000000029          893.0          741.0         1016.0
cg000000103           NaN           NaN           NaN
cg000000109         2710.0         1930.0         2029.0
cg000000155         4403.0         4302.0         3865.0
cg000000158         5678.0         4738.0         4938.0

      203175700025_R02C01  203175700025_R03C01  203175700025_R04C01  \
IlmnID
cg000000029         1272.0         1035.0         1423.0
cg000000103           NaN           NaN           NaN
cg000000109         2551.0         2385.0         3010.0
cg000000155         3734.0         4996.0         5297.0
cg000000158         4690.0         5658.0         6767.0

      203175700025_R05C01  203175700025_R06C01  203175700025_R07C01  \
IlmnID
cg000000029          917.0          904.0          743.0
cg000000103           NaN           NaN           NaN
cg000000109         2219.0         1716.0         2132.0
cg000000155         3787.0         5216.0         4389.0
cg000000158         6030.0         5780.0         6300.0

      203175700025_R08C01
IlmnID
cg000000029         1386.0
cg000000103           NaN
cg000000109         2158.0
cg000000155         4404.0
cg000000158         5050.0
```

```
[4]: # noob-normalized probe values
(noob_meth,noob_unmeth) = methylcheck.load(filepath, format='noob_df')
```

(continues on next page)

(continued from previous page)

```
noob_meth.head()
```

```
100%| 16/16 [00:00<00:00, 550.34it/s]
100%| 16/16 [00:00<00:00, 1499.81it/s]
INFO:methylcheck.load_processed:(865859, 16), (865859, 16)
```

```
[4]:      203163220027_R01C01  203163220027_R02C01  203163220027_R03C01  \
IlmnID
cg000000029      1174.0      1123.0      973.0
cg000000103      NaN      NaN      NaN
cg000000109      3372.0      3003.0      2728.0
cg000000155      4647.0      5488.0      4754.0
cg000000158      7810.0      7100.0      6721.0

      203163220027_R04C01  203163220027_R05C01  203163220027_R06C01  \
IlmnID
cg000000029      NaN      1745.0      1355.0
cg000000103      NaN      NaN      NaN
cg000000109      3645.0      2950.0      2766.0
cg000000155      4858.0      5390.0      5019.0
cg000000158      5904.0      6912.0      7875.0

      203163220027_R07C01  203163220027_R08C01  203175700025_R01C01  \
IlmnID
cg000000029      1074.0      812.0      1342.0
cg000000103      NaN      NaN      NaN
cg000000109      3160.0      2185.0      2684.0
cg000000155      5198.0      5042.0      5300.0
cg000000158      6893.0      5625.0      6925.0

      203175700025_R02C01  203175700025_R03C01  203175700025_R04C01  \
IlmnID
cg000000029      1497.0      1286.0      1696.0
cg000000103      NaN      NaN      NaN
cg000000109      3191.0      2902.0      3526.0
cg000000155      4875.0      6244.0      6430.0
cg000000158      6278.0      7188.0      8478.0

      203175700025_R05C01  203175700025_R06C01  203175700025_R07C01  \
IlmnID
cg000000029      1018.0      956.0      820.0
cg000000103      NaN      NaN      NaN
cg000000109      2488.0      1883.0      2384.0
cg000000155      4254.0      6116.0      4892.0
cg000000158      7100.0      6859.0      7247.0

      203175700025_R08C01
IlmnID
cg000000029      1524.0
cg000000103      NaN
cg000000109      2368.0
cg000000155      5048.0
cg000000158      5887.0
```

```
[5]: # m_values
m_values = methylcheck.load(filepath, format='m_value')
m_values.head()
```

```
Files: 100%|| 1/1 [00:00<00:00, 5.41it/s]
INFO:methylcheck.load_processed:loaded data (865859, 16) from 1 pickled files (0.22s)
```

```
[5]:
      203163220027_R01C01  203163220027_R02C01  203163220027_R03C01  \
IlmnID
cg000000029          2.525          1.579          1.500
cg000000103          NaN          NaN          NaN
cg000000109          4.047          3.444          2.930
cg000000155          4.582          4.681          4.528
cg000000158          4.960          5.135          5.042

      203163220027_R04C01  203163220027_R05C01  203163220027_R06C01  \
IlmnID
cg000000029          NaN          3.028          3.109
cg000000103          NaN          NaN          NaN
cg000000109          4.269          3.875          1.775
cg000000155          4.518          4.717          4.538
cg000000158          4.928          4.909          4.748

      203163220027_R07C01  203163220027_R08C01  203175700025_R01C01  \
IlmnID
cg000000029          1.643          1.466          2.055
cg000000103          NaN          NaN          NaN
cg000000109          3.780          3.516          3.133
cg000000155          4.797          4.960          4.564
cg000000158          4.996          5.238          4.746

      203175700025_R02C01  203175700025_R03C01  203175700025_R04C01  \
IlmnID
cg000000029          2.890          1.508          2.940
cg000000103          NaN          NaN          NaN
cg000000109          3.912          3.748          4.358
cg000000155          4.444          4.738          4.606
cg000000158          4.958          4.568          4.920

      203175700025_R05C01  203175700025_R06C01  203175700025_R07C01  \
IlmnID
cg000000029          1.175          1.819          0.685
cg000000103          NaN          NaN          NaN
cg000000109          3.162          3.008          1.468
cg000000155          4.547          4.885          4.641
cg000000158          4.788          4.795          5.284

      203175700025_R08C01
IlmnID
cg000000029          3.027
cg000000103          NaN
cg000000109          3.045
cg000000155          4.567
cg000000158          4.844
```

**Note:** If you point to a folder with multiple batches of samples, of different array types, you'll get an error. All samples in the path you choose need to have the same number of probes (same array type). Here is the message you'll get in this scenario.

```
ValueError: all the input array dimensions for the concatenation axis must match_
↳ exactly,
but along dimension 0, the array at index 0 has size 226618 and the array at index 1_
↳ has size 244827 (continues on next page)
```

(continued from previous page)

This shouldn't happen if you download and process data using `methyprep` (which will automatically detect array types and create separate folders for each array type). If you didn't process with `methyprep`, the workaround is manually moving the files from different array-types into separate folders and loading them separately.

The `meth` format for `methycheck.load()` returns a list of `SampleDataContainer` objects.

`sd[0]._SampleDataContainer__data_frame` will give you a dataframe of the first `SampleDataContainer` in your list called `sd`.

### 4.1.3 Loading Data from .csv Files

`methycheck` assumes you want to load data the fastest way, using a single high-performance python3 pickled dataframe. But there are times when you want to load from CSV output files instead. One use case is where you want to examine probes that were filtered out by `poobah` (p-value probe detection). The CSVs contain this information, whereas the pickled dataframe has it removed by default.

If you wish to load the beta values from CSVs, point the function to the parent directory where your CSVs are and it will automatically go through that directory recursively to concatenate all of the beta value columns from each CSV present. Please note that this will take longer than reading it from the beta pickle.

This function will by default mask the beta values of failed probes with `NaN` ( $p \leq 0.05$ ). If you wish to view your raw beta values without any influence by `poobah`, add the argument `no_poobah=True`.

```
[6]: df = methycheck.load(filepath, format='beta_csv')
df

Files: 0%|          | 0/16 [00:00<?, ?it/s]INFO:numexpr.utils:NumExpr defaulting to_
↳8 threads.
Files: 100%|| 16/16 [00:20<00:00, 1.29s/it]
INFO:methycheck.load_processed:merging...
100%|| 16/16 [00:00<00:00, 524.09it/s]
```

```
[6]:
```

	203163220027_R02C01	203163220027_R06C01	203163220027_R08C01	\
IlmnID				
cg00000029	0.749	0.896	0.734	
cg00000103	0.922	0.950	0.654	
cg00000109	0.916	0.774	0.920	
cg00000155	0.962	0.959	0.969	
cg00000158	0.972	0.964	0.974	
...	...	...	...	
rs9363764	0.544	0.544	0.592	
rs939290	0.055	0.591	0.974	
rs951295	0.557	0.032	0.566	
rs966367	0.966	0.468	0.028	
rs9839873	0.968	0.618	0.975	

	203163220027_R03C01	203163220027_R07C01	203163220027_R04C01	\
IlmnID				
cg00000029	0.739	0.757	NaN	
cg00000103	0.931	0.712	0.924	
cg00000109	0.884	0.932	0.951	
cg00000155	0.958	0.965	0.958	
cg00000158	0.971	0.970	0.968	
...	...	...	...	
rs9363764	0.062	0.058	0.431	

(continues on next page)

(continued from previous page)

rs939290	0.051	0.965	0.809
rs951295	0.515	0.955	0.966
rs966367	0.956	0.499	0.036
rs9839873	0.642	0.671	0.962
203163220027_R01C01 203163220027_R05C01 203175700025_R02C01 \			
IlmnID			
cg00000029	0.852	0.891	0.881
cg00000103	0.943	0.931	0.943
cg00000109	0.943	0.936	0.938
cg00000155	0.960	0.963	0.956
cg00000158	0.969	0.968	0.969
...	...	...	...
rs9363764	0.962	0.036	0.046
rs939290	0.966	0.070	0.964
rs951295	0.548	0.975	0.024
rs966367	0.029	0.478	0.488
rs9839873	0.969	0.969	0.969
203175700025_R06C01 203175700025_R08C01 203175700025_R03C01 \			
IlmnID			
cg00000029	0.779	0.891	0.740
cg00000103	0.920	0.731	0.809
cg00000109	0.889	0.892	0.931
cg00000155	0.967	0.960	0.964
cg00000158	0.965	0.966	0.960
...	...	...	...
rs9363764	0.562	0.046	0.054
rs939290	0.578	0.601	0.535
rs951295	0.073	0.556	0.035
rs966367	0.483	0.526	0.034
rs9839873	0.613	0.669	0.967
203175700025_R07C01 203175700025_R04C01 203175700025_R01C01 \			
IlmnID			
cg00000029	0.617	0.885	0.806
cg00000103	0.611	0.365	0.946
cg00000109	0.734	0.953	0.898
cg00000155	0.961	0.961	0.959
cg00000158	0.975	0.968	0.964
...	...	...	...
rs9363764	0.541	0.551	0.971
rs939290	0.625	0.603	0.962
rs951295	0.974	0.523	0.029
rs966367	0.455	0.034	0.027
rs9839873	0.974	0.972	0.970
203175700025_R05C01			
IlmnID			
cg00000029	0.693		
cg00000103	0.934		
cg00000109	0.899		
cg00000155	0.959		
cg00000158	0.965		
...	...		
rs9363764	0.966		
rs939290	0.965		

(continues on next page)

(continued from previous page)

```
rs951295          0.949
rs966367          0.051
rs9839873         0.050
```

```
[865918 rows x 16 columns]
```

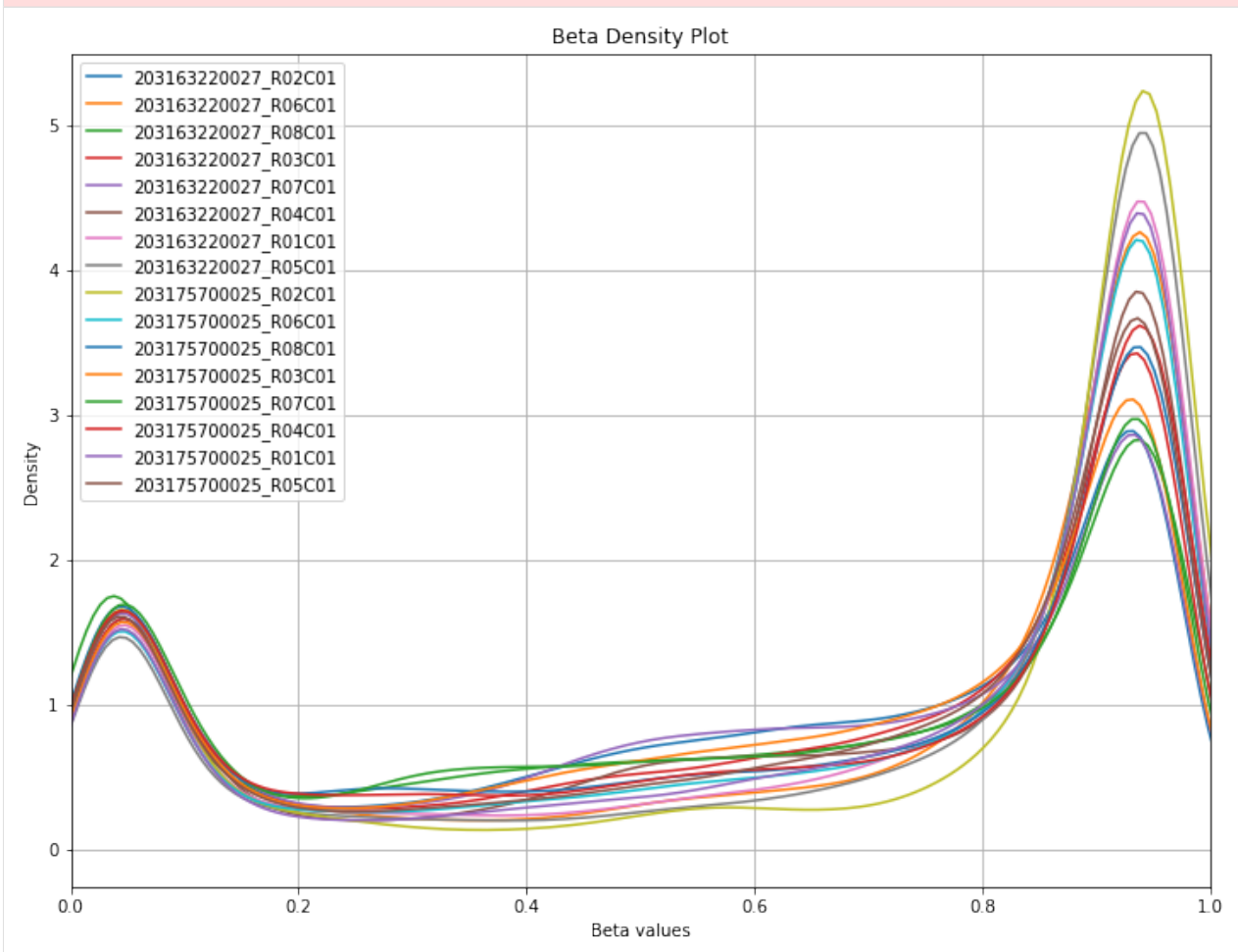
#### 4.1.4 Checking Beta Distributions

One of the first steps users should take when loading their data is examining the beta distribution and ensuring that the expected bimodal distribution is present. Samples whose beta distributions don't fit the bimodal distribution are more likely to be poor quality samples that need to be filtered out (you may be expecting a different distribution based on what kind of data you're looking at, of course, but the standard is bimodal with peaks around 0 and 1).

`methyIcheck` includes a beta density plot function that is similar to `seaborn`'s `kde` plot. If the sample size is small enough (<30), sample names will be included in the legend of the plot, so you may identify outlier samples easily.

```
[7]: methyIcheck.beta_density_plot(df)
```

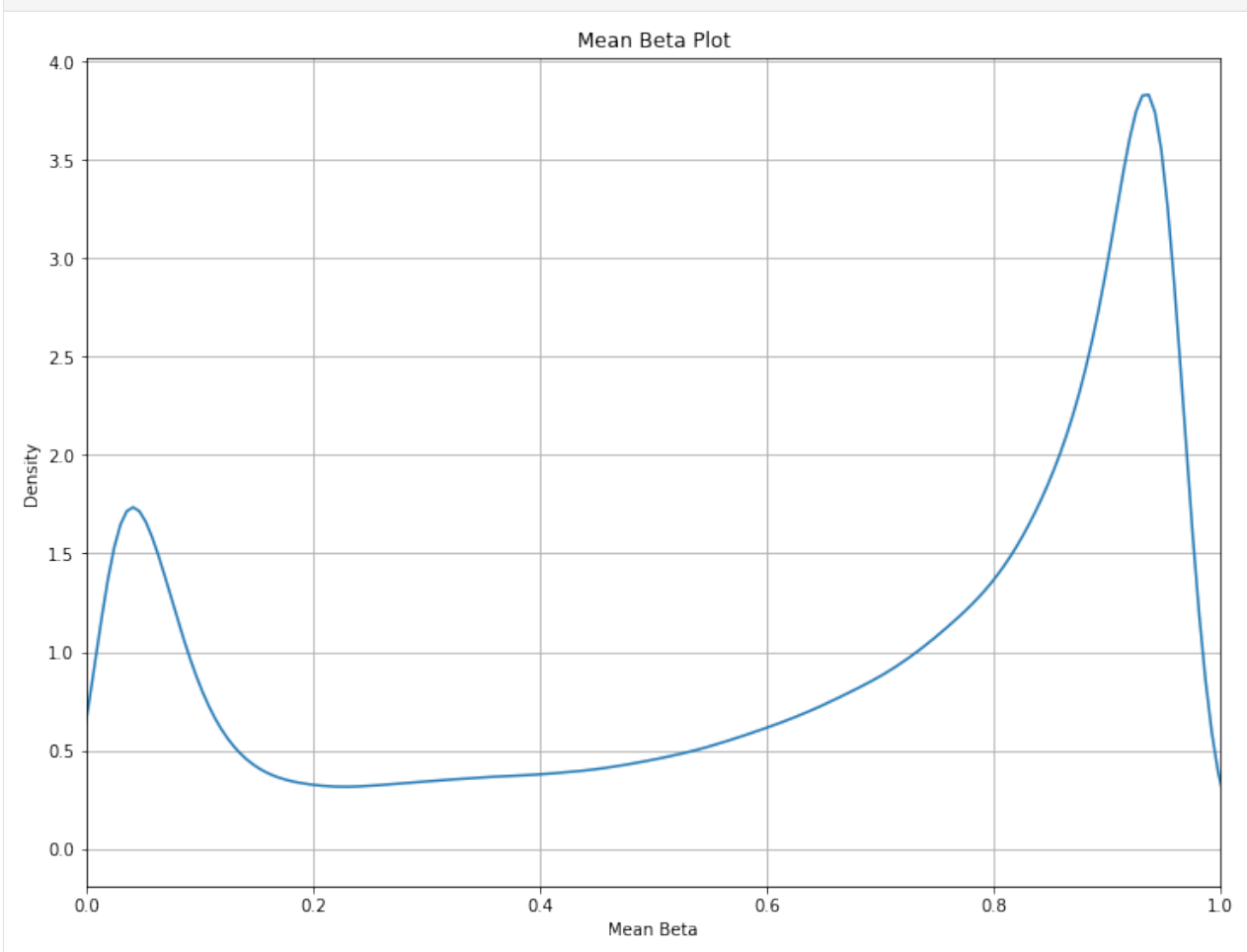
```
WARNING:methyIcheck.samples.postprocessQC:Your data contains 7585 missing probe_
↪ values per sample, (121361 overall). For a list per sample, use verbose=True
```



This data looks relatively clean! Most data on GEO should be high quality, but it always pays to check.

You may also want to get an idea of the mean beta distribution (useful for comparing multiple datasets, or for looking at a dataset before/after quality control measures.) Check the filtering probes section for examples on comparing beta distributions and what filtering can do to your data’s mean beta distribution.

```
[8]: methycheck.mean_beta_plot(df)
```



## 4.2 Filtering Poor Quality Probes

Note: Separate from any filtering that `methycheck` does, the `qualityMask` step of the `methy prep` processing pipeline excludes a list of probes that are historically poor quality (`SeSAmE` masks the same list of probes, which are from the Zhou 2016 paper [linked below]). This can be turned off in `run_pipeline` by specifying `quality_mask=False` if desired. There are several criteria for exclusion of probes that `methycheck` offers. These are designed to be in line with past research that has identified “sketchy” probes. Areas that have polymorphisms, cross-hybridization, repeat sequence elements, or base color changes can affect probe quality. `methycheck`’s `list_problem_probes()` function returns a list of probes excluded based on literature. For each array type, the publication list is as follows:

- 450k: ‘Chen2013’, ‘Price2013’, ‘Zhou2016’, ‘Naeem2014’, ‘DacaRoszak2015’
- EPIC: ‘Zhou2016’, ‘McCartney2016’

The articles are linked above, for any users that would like more detail on what probes are considered problematic and what criteria the authors have used to identify them. The lists are also formatted in the acceptable input style for the `list_problem_probes` function. See below for examples of its usage.



```
[1]: import methylcheck
      from pathlib import Path
      filepath = Path('/Users/patriciagirardi/tutorial/GPL21145')
```

### 4.2.1 Available probe exclusion lists

We’ve imported lists of methylation probes that various researchers have previously deemed to be “sketchy.” You can use methylcheck to remove these probes – by referring to the list by the publication’s first author name, or by the reason these probes should be excluded from analysis.

```
[2]: # this code will print the criteria reason (either a publication or a type of issue,
      ↪ like Polymorphism)
      # as well as the number of probes excluded for that reason

      criteria = ['Chen2013', 'Price2013', 'Naeem2014', 'DacaRoszak2015', 'Polymorphism',
                  'CrossHybridization', 'BaseColorChange', 'RepeatSequenceElements']
      EPIC_criteria = ['McCartney2016', 'Zhou2016', 'Polymorphism', 'CrossHybridization',
                      ↪ 'BaseColorChange', 'RepeatSequenceElements']

      print('450k probe exclusion criteria and number of probes excluded:')
      for crit in criteria:
          print(crit, '--', len(methylcheck.list_problem_probes('450k', [crit])))

      print('\nEPIC probe exclusion criteria and number of probes excluded:')
      for crit in EPIC_criteria:
          print(crit, '--', len(methylcheck.list_problem_probes('EPIC', [crit])))

      450k probe exclusion criteria and number of probes excluded:
      Chen2013 -- 265410
      Price2013 -- 213246
      Naeem2014 -- 128695
      DacaRoszak2015 -- 89678
      Polymorphism -- 289952
      CrossHybridization -- 92524
      BaseColorChange -- 359
      RepeatSequenceElements -- 96631

      EPIC probe exclusion criteria and number of probes excluded:
      McCartney2016 -- 326267
      Zhou2016 -- 178671
      Polymorphism -- 346033
      CrossHybridization -- 108172
      BaseColorChange -- 406
      RepeatSequenceElements -- 0
```

```
[3]: # users may also get the list of probe names that are excluded for any of the criteria
      methylcheck.list_problem_probes(array='epic', criteria=['Polymorphism'])[0:5]
```

```
[3]: ['cg14670079', 'cg26778521', 'cg04785903', 'cg15989966', 'cg21932343']
```

### 4.2.2 Filtering poor quality probes

```
[4]: # leave criteria undefined to list all problem probes for that array type
      sketchy_probes_list = methylcheck.list_problem_probes(array='epic')
```

```
[5]: df = methylcheck.load(filepath)

# with the sketchy_probes_list defined above, we can use methylcheck.exclude_probes()
↳ to remove all the unwanted probes
excluded_df = methylcheck.exclude_probes(df, sketchy_probes_list)
excluded_df.head()

Files: 100%| 1/1 [00:00<00:00, 4.75it/s]
INFO:methylcheck.load_processed:loaded data (865859, 16) from 1 pickled files (0.253s)

Of {len(df.index)} probes, {post_overlap} matched, yielding {len(df.index)-post_
↳ overlap} probes after filtering.
```

```
[5]:
```

	203163220027_R01C01	203163220027_R02C01	203163220027_R03C01	\
IlmnID				
cg000000029	0.852	0.749	0.739	
cg000000109	0.943	0.916	0.884	
cg000000165	0.237	0.733	0.374	
cg000000221	0.943	0.937	0.939	
cg000000236	0.922	0.934	0.919	

	203163220027_R04C01	203163220027_R05C01	203163220027_R06C01	\
IlmnID				
cg000000029	NaN	0.891	0.896	
cg000000109	0.951	0.936	0.774	
cg000000165	0.639	0.397	0.564	
cg000000221	0.926	0.942	0.945	
cg000000236	0.881	0.926	0.882	

	203163220027_R07C01	203163220027_R08C01	203175700025_R01C01	\
IlmnID				
cg000000029	0.757	0.734	0.806	
cg000000109	0.932	0.920	0.898	
cg000000165	0.486	0.194	0.163	
cg000000221	0.938	0.933	0.949	
cg000000236	0.927	0.774	0.937	

	203175700025_R02C01	203175700025_R03C01	203175700025_R04C01	\
IlmnID				
cg000000029	0.881	0.740	0.885	
cg000000109	0.938	0.931	0.953	
cg000000165	0.557	0.531	0.856	
cg000000221	0.936	0.934	0.949	
cg000000236	0.957	0.932	0.952	

	203175700025_R05C01	203175700025_R06C01	203175700025_R07C01	\
IlmnID				
cg000000029	0.693	0.779	0.617	
cg000000109	0.899	0.889	0.734	
cg000000165	0.171	0.173	0.865	
cg000000221	0.944	0.937	0.942	
cg000000236	0.923	0.938	0.932	

	203175700025_R08C01
IlmnID	
cg000000029	0.891
cg000000109	0.892
cg000000165	0.581
cg000000221	0.934

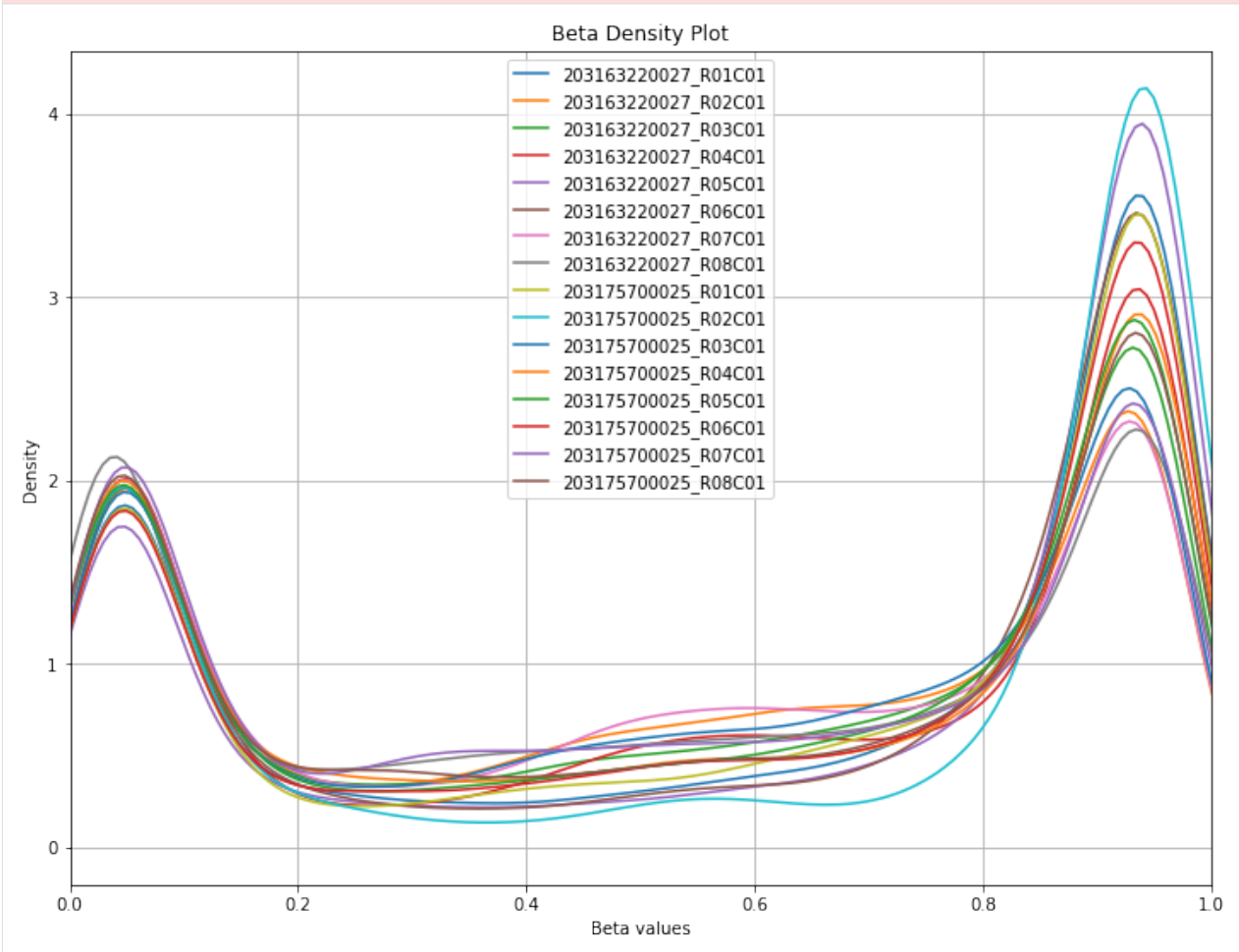
(continues on next page)

(continued from previous page)

cg00000236 0.951

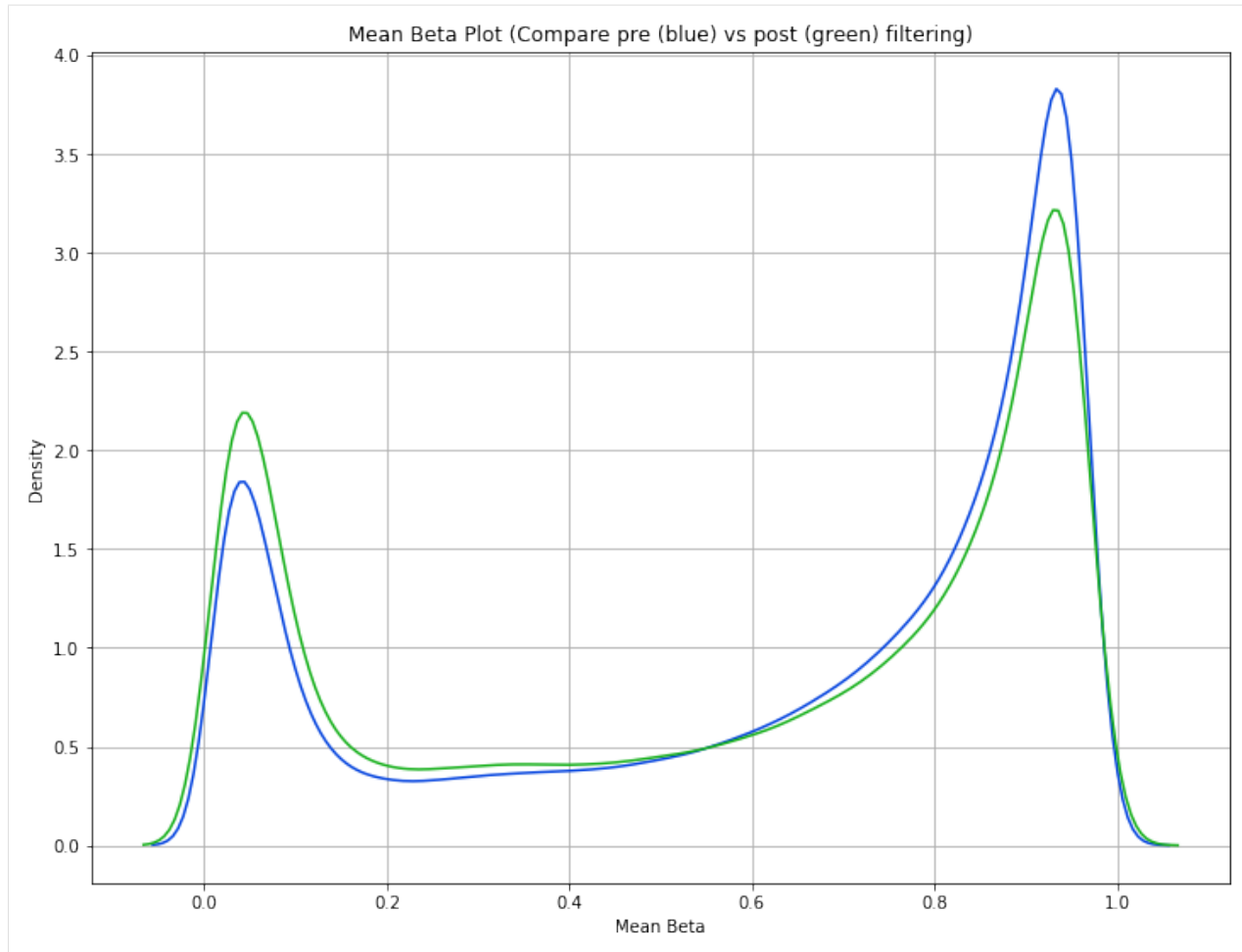
```
[6]: methylcheck.beta_density_plot(excluded_df)
```

WARNING:methylcheck.samples.postprocessQC:Your data contains 6103 missing probe\_ values per sample, (97656 overall). For a list per sample, use verbose=True  
INFO:numexpr.utils:NumExpr defaulting to 8 threads.



To get a sense for how this affects the data, users may want to compare the mean beta density distribution before and after probe filtering.

```
[7]: methylcheck.mean_beta_compare(df, excluded_df)
```



### 4.2.3 Filtering sex-linked probes and control probes

Other probe types that are often filtered out are sex-linked probes and quality control probes used by Illumina. Quality control probes are automatically filtered out in `methyprep` processing with the default `qualityMask`, so there's no need to run `exclude_sex_control_probes` if you processed your data with `methyprep`. Otherwise, we recommend `methycheck`'s `exclude_sex_control_probes` to remove both sex-linked probes and quality control probes.

```
[8]: filtered_df = methycheck.exclude_sex_control_probes(excluded_df, 'epic', no_sex=True,
→ no_control=True, verbose=True)
```

```
epic: Removed 12360 sex-linked probes from 16 samples. 464871 probes remaining.
```

```
[9]: filtered_df.head()
```

```
[9]:
```

	203163220027_R01C01	203163220027_R02C01	203163220027_R03C01	\
IlmnID				
cg000000029	0.852	0.749	0.739	
cg000000109	0.943	0.916	0.884	
cg000000165	0.237	0.733	0.374	
cg000000221	0.943	0.937	0.939	
cg000000236	0.922	0.934	0.919	

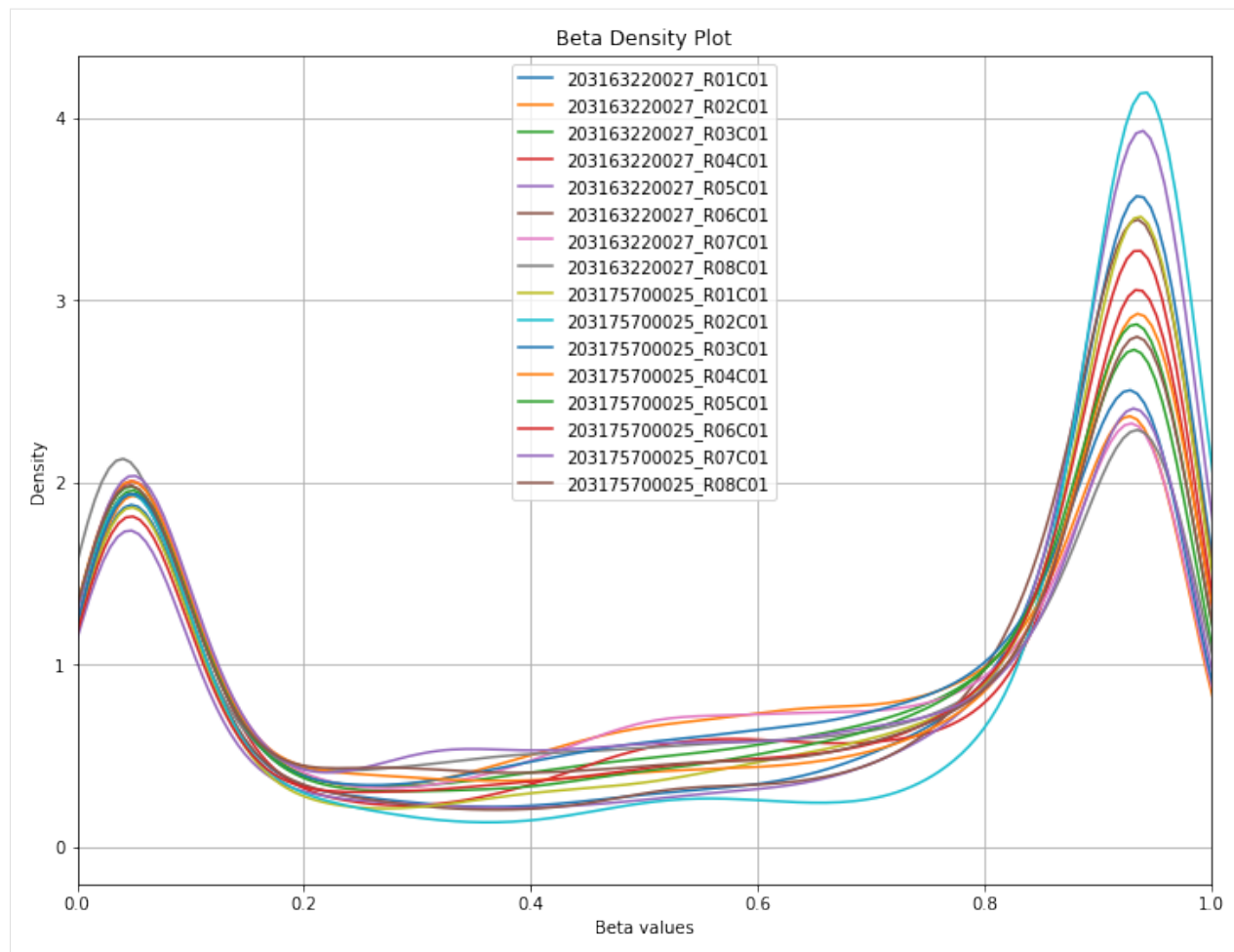
(continues on next page)

(continued from previous page)

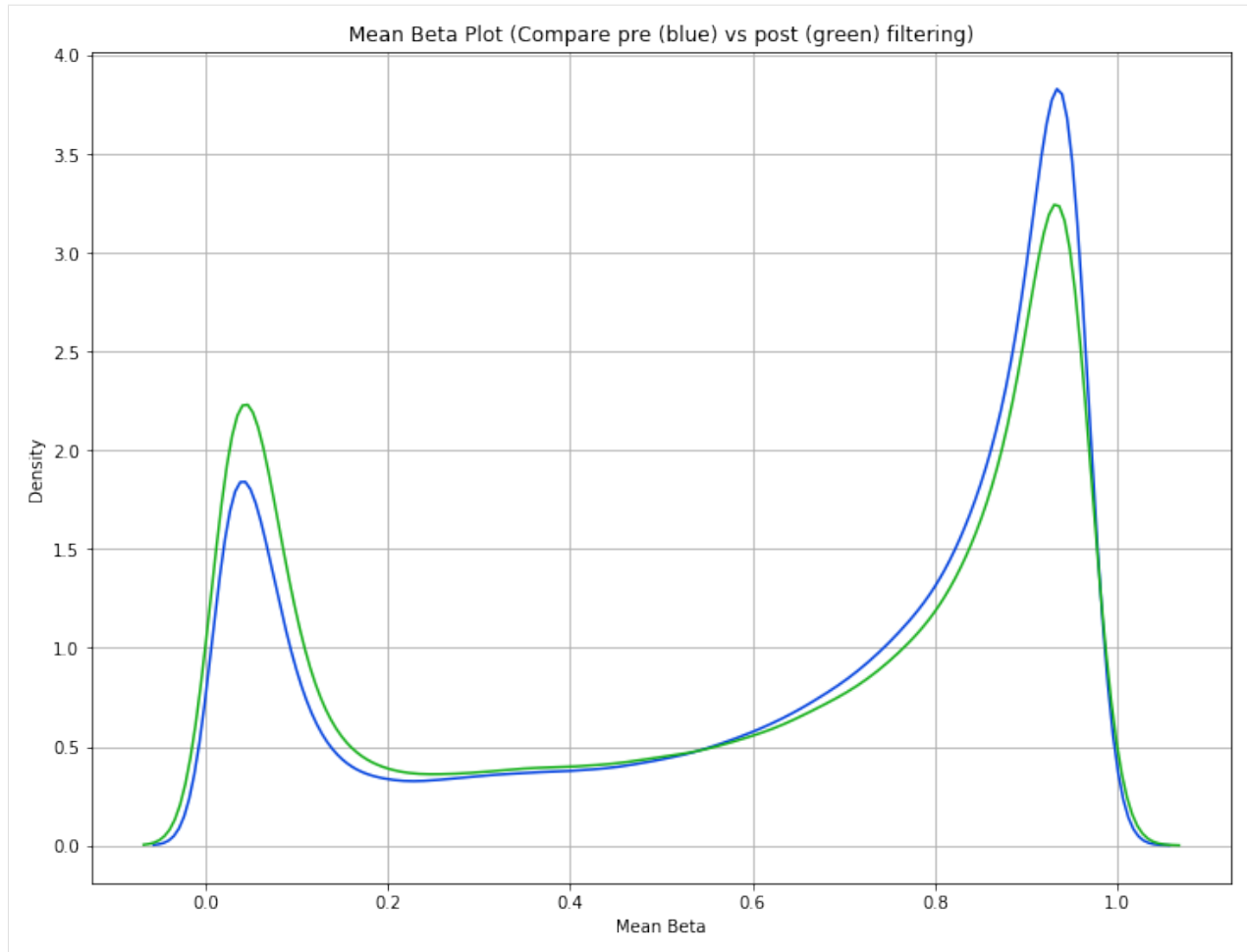
	203163220027_R04C01	203163220027_R05C01	203163220027_R06C01	\
IlmnID				
cg000000029	NaN	0.891	0.896	
cg000000109	0.951	0.936	0.774	
cg000000165	0.639	0.397	0.564	
cg000000221	0.926	0.942	0.945	
cg000000236	0.881	0.926	0.882	
	203163220027_R07C01	203163220027_R08C01	203175700025_R01C01	\
IlmnID				
cg000000029	0.757	0.734	0.806	
cg000000109	0.932	0.920	0.898	
cg000000165	0.486	0.194	0.163	
cg000000221	0.938	0.933	0.949	
cg000000236	0.927	0.774	0.937	
	203175700025_R02C01	203175700025_R03C01	203175700025_R04C01	\
IlmnID				
cg000000029	0.881	0.740	0.885	
cg000000109	0.938	0.931	0.953	
cg000000165	0.557	0.531	0.856	
cg000000221	0.936	0.934	0.949	
cg000000236	0.957	0.932	0.952	
	203175700025_R05C01	203175700025_R06C01	203175700025_R07C01	\
IlmnID				
cg000000029	0.693	0.779	0.617	
cg000000109	0.899	0.889	0.734	
cg000000165	0.171	0.173	0.865	
cg000000221	0.944	0.937	0.942	
cg000000236	0.923	0.938	0.932	
	203175700025_R08C01			
IlmnID				
cg000000029	0.891			
cg000000109	0.892			
cg000000165	0.581			
cg000000221	0.934			
cg000000236	0.951			

```
[10]: methylcheck.beta_density_plot(filtered_df)
```

```
WARNING:methylcheck.samples.postprocessQC:Your data contains 5626 missing probe_
↪values per sample, (90031 overall). For a list per sample, use verbose=True
```



```
[11]: methylcheck.mean_beta_compare(df, filtered_df)
```



## 4.3 Quality Control

### 4.3.1 `controls_report` (a color-coded spreadsheet of control probe performance per sample)

`methyIcheck` is geared toward quality control of processed data. To this end, there is a helpful function that summarizes performance of control probes ([details on control probes here](#)). To run this function, the `control_probes.pkl` file output from `methyIprep` is required. This report ensures that the chemistry (bisulfite conversion, target specificity, hybridization, staining, etc.) and machine readings are acceptable.

There is an optional portion of this report that relies on values from the poobah file as well. If no poobah file is present, this part is ignored.

Check the guide linked above for more information on how to read these reports. They're intuitively color-coded (green = passing, red = failing, yellow = somewhere in between) so they're easy to read at a glance. There is a colorblind-friendly option included in this function.

We'll walk through examples of the QC pipeline using pre-processed, filtered data (see the [Filtering Probes](#) page).

```
[1]: import methylcheck
      from pathlib import Path
      filepath = Path('/Users/patriciagirardi/tutorial/GPL21145')
```

```
[2]: df, metadata = methylcheck.load_both(filepath=filepath)
      metadata.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:['/Users/patriciagirardi/tutorial/\_  
 ↳GPL21145/GSE147391\_GPL21145\_meta\_data.pkl', '/Users/patriciagirardi/tutorial/\_  
 ↳GPL21145/sample\_sheet\_meta\_data.pkl']  
 INFO:methylcheck.load\_processed:Multiple meta\_data found. Only loading the first file.  
 INFO:methylcheck.load\_processed:Loading 16 samples.  
 Files: 100%| 1/1 [00:00<00:00, 5.07it/s]  
 INFO:methylcheck.load\_processed:loaded data (865859, 16) from 1 pickled files (0.223s)  
 INFO:methylcheck.load\_processed:meta.Sample\_IDs match data.index (OK)

```
[2]:
```

	GSM_ID	Sample_Name	Sentrix_ID	Sentrix_Position	\
2	GSM4429898	Grade II rep3	203163220027		R01C01
3	GSM4429899	Grade III rep1	203163220027		R02C01
12	GSM4429908	Grade IV rep5	203163220027		R03C01
15	GSM4429911	Grade IV rep8	203163220027		R04C01
6	GSM4429902	Grade II rep6	203163220027		R05C01

	source	histological diagnosis	description	\
2	Resected glioma	Diffuse astrocytoma (II)		Glioma
3	Resected glioma	Anaplastic astrocytoma (III)		Glioma
12	Resected glioma	Glioblastoma (IV)		Glioma
15	Resected glioma	Glioblastoma (IV)		Glioma
6	Resected glioma	Oligodendroglioma (II)		Glioma

	Sample_ID
2	203163220027_R01C01
3	203163220027_R02C01
12	203163220027_R03C01
15	203163220027_R04C01
6	203163220027_R05C01

```
[3]: methylcheck.controls_report(filepath=filepath)
```

INFO:methylprep.files.manifests:Reading manifest file: MethylationEPIC\_v1-0\_B4.  
 ↳CoreColumns.csv

```
203175700025_R01C01 GA r=0.81 ±0.23 p<0.0 |CT r=0.82 ±0.15 p<0.0
203175700025_R02C01 GA r=0.9 ±0.16 p<0.0 |CT r=0.87 ±0.13 p<0.0
203163220027_R01C01 GA r=0.86 ±0.19 p<0.0 |CT r=0.83 ±0.14 p<0.0
203163220027_R02C01 GA r=0.88 ±0.16 p<0.0 |CT r=0.85 ±0.13 p<0.0
203175700025_R05C01 GA r=0.87 ±0.14 p<0.0 |CT r=0.84 ±0.11 p<0.0
203175700025_R06C01 GA r=0.91 ±0.12 p<0.0 |CT r=0.87 ±0.1 p<0.0
203163220027_R05C01 GA r=0.87 ±0.16 p<0.0 |CT r=0.85 ±0.12 p<0.0
203163220027_R06C01 GA r=0.9 ±0.14 p<0.0 |CT r=0.85 ±0.12 p<0.0
203175700025_R03C01 GA r=0.85 ±0.17 p<0.0 |CT r=0.85 ±0.13 p<0.0
203175700025_R04C01 GA r=0.87 ±0.15 p<0.0 |CT r=0.87 ±0.11 p<0.0
203175700025_R07C01 GA r=0.88 ±0.13 p<0.0 |CT r=0.85 ±0.11 p<0.0
203175700025_R08C01 GA r=0.9 ±0.13 p<0.0 |CT r=0.87 ±0.1 p<0.0
203163220027_R03C01 GA r=0.84 ±0.17 p<0.0 |CT r=0.83 ±0.13 p<0.0
```

(continues on next page)



(continued from previous page)

```
203163220027_R07C01 GA r=0.85 ±0.16 p<0.0 |CT r=0.86 ±0.11 p<0.0
203163220027_R08C01 GA r=0.87 ±0.15 p<0.0 |CT r=0.85 ±0.12 p<0.0

INFO:methylcheck.reports.controls_report:Predicting Sex...
INFO:methylprep.files.manifests:Reading manifest file: MethylationEPIC_v-1-0_B4.
→CoreColumns.csv

203163220027_R04C01 GA r=0.87 ±0.16 p<0.0 |CT r=0.87 ±0.12 p<0.0
(865859, 16) (865859, 16)
```

The color-coded results are contained in an excel file that will be saved in the same directory that was specified as an input. It will look similar to this:

	R	S	T	U	V	W	X	Y	Z	
1	Red	Specificity II	Specificity II Bkg	Non-polymorphic Green	Non-polymorphic Red	Baseline Green	Baseline Red	Negative Baseline G	Negative Baseline R	Reg
2	h	(S_Red/S_Green)	(background/S_Green)	(min(CG)/max(AT))	(min(AT)/max(CG))	max[Extension (A), Extension (T)]	max[Extension (C), Extension (G)]	mean NEGATIVE Green control probes	mean NEGATIVE Red control probes	NOR
3	max(MM) > 1	> 1.0	> 1.0	> 5.0	> 5.0	no offset	no offset			NOR
4	7.96	41.68	12.8	13.67	26.27	175	970	89	91	
5	6.78	53.07	15.43	12.82	19.61	210	1142	107	119	
6	8.38	43.4	13.33	11.15	25.16	185	946	94	93	
7	7.78	41.04	12.23	12.16	21.42	204	925	100	92	
8	7.77	30.92	10.2	13.68	28.84	223	1080	103	86	
9	6.94	33.98	9.52	14.67	18.15	181	1046	105	93	
10	7.27	36.4	12.4	10.9	20.98	248	1068	109	116	
11	7.23	36.62	12.69	8.47	22.92	198	990	117	96	
12	7.41	37.88	11.13	12.15	24.16	217	1026	104	93	
13	7.02	42.41	11.91	12.62	24.46	217	977	111	81	
14	7.51	35.36	10.99	12.55	24.76	208	938	117	83	
15	7.41	44.58	13.71	12.95	22.15	249	1134	119	111	
16	7.04	43.11	12.98	11.79	22.17	193	930	113	109	
17	8.07	39.03	12.03	11.7	24.59	236	1032	118	83	
18	9.67	40.27	12.65	14.19	27.72	213	1088	120	107	
19	6.93	35.51	10.65	12.18	20.66	194	1021	108	95	

The “Passing Probes” column is the column that relies on values from the poobah file. This is a measure of how many probes failed in each sample (detection p-value > 0.05). The p-value cut off is adjustable with the `pval_sig` argument, which is set to 0.05 by default.

Notice the final column (“Result”) where most samples are passing. This column is calculated by checking that all of the QC columns are above a minimum threshold. This threshold is adjustable with the `passing` argument (set to 0.7 by default). - If the poobah file is included and 20% or more of probes fail, the Result is automatically FAIL. - If 70% of the columns are passing, the result is “OK” or passing. - If more than 70% are passing, but less than 100%, the “OK” will have a number next to it to specify what percentage of columns passed. - If less than 70% of the columns passed, the result is either FAIL or MARGINAL (based on how close to the 70% threshold it got).

The predicted sex column is based on the median values of methylation measurements on the X and Y chromosomes. If `yMed - xMed` is less than whatever the specified cutoff value is, it is predicted female. Otherwise, the predicted sex is male. Also, for samples from female subjects, at least 90% of the Y chromosome probes should fail. If there is a sample sheet that includes “sex” or “gender”, the reporter will also flag any mismatches between the predicted sex and the specified sex (this does not affect the Result column).

### 4.3.2 Quality Control in the IDE

If you want to run a quick quality control check within your CLI or IDE (with no output files saved), use the `run_qc` function. This is a simplified version of the full `controls_report` function.

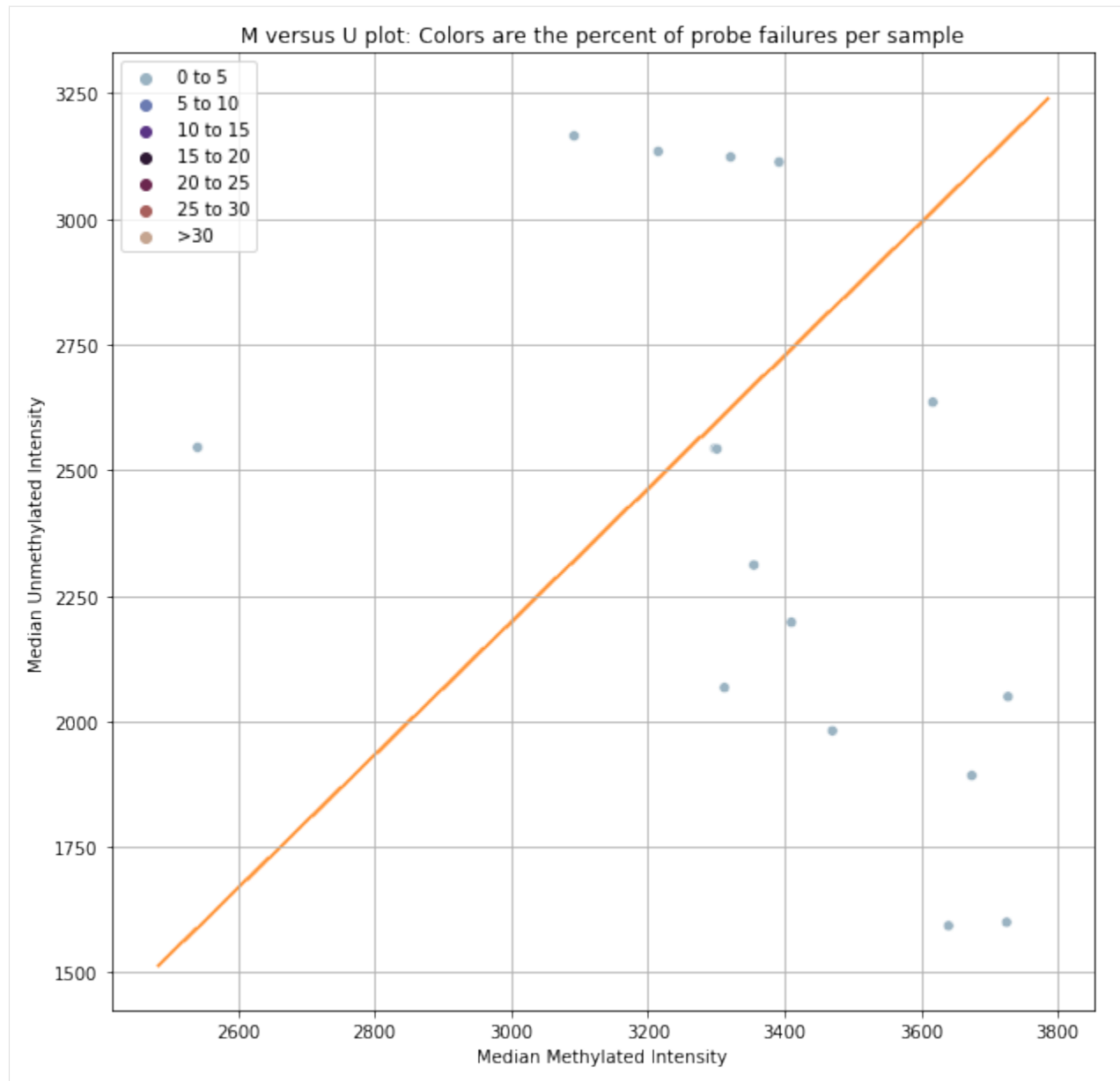
The first and second plots shown below are based on the `qc_signal_intensity` function, which suggests sample outliers based on methylated and unmethylated signal intensity. The cutoff value is based on `minfi`'s chosen cutoff value. If a sample falls below the dotted line in the second chart, it is potentially poor quality (due to low fluorescence). This is a very specific way of identifying poor quality samples and we recommend taking a more holistic approach (using `controls_report()`). However, this can still be useful information in some circumstances, which is why we've included it in the original qc pipeline.

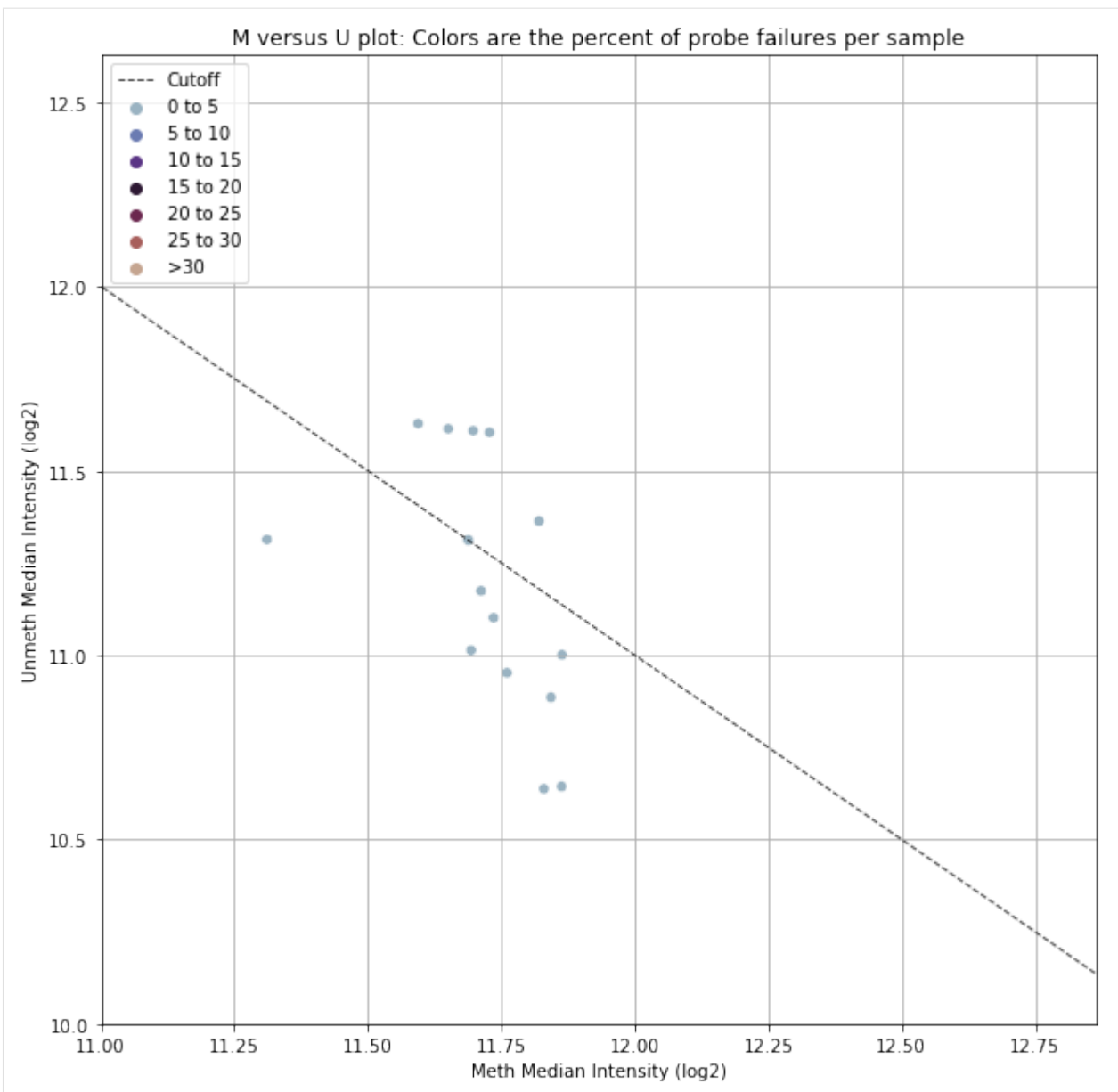
The next set of plots cover hybridization, staining, bisulfite conversion, specificity, target removal, extension, negative controls, and non-polymorphic binding, but can be more difficult to interpret in this graph format. For more, consult [Illumina's technical document](#) on what the expected values for these graphs should be.

The final set of plots is beta distributions by probe type (Type I and Type II); the distributions of Type I probes are split by color channel as well.

This function is to be used before any removal of probes or samples. If you remove probes or samples from your beta value dataframe, this function will no longer run due to inconsistent rows and/or columns in other dataframes (such as `m-value`).

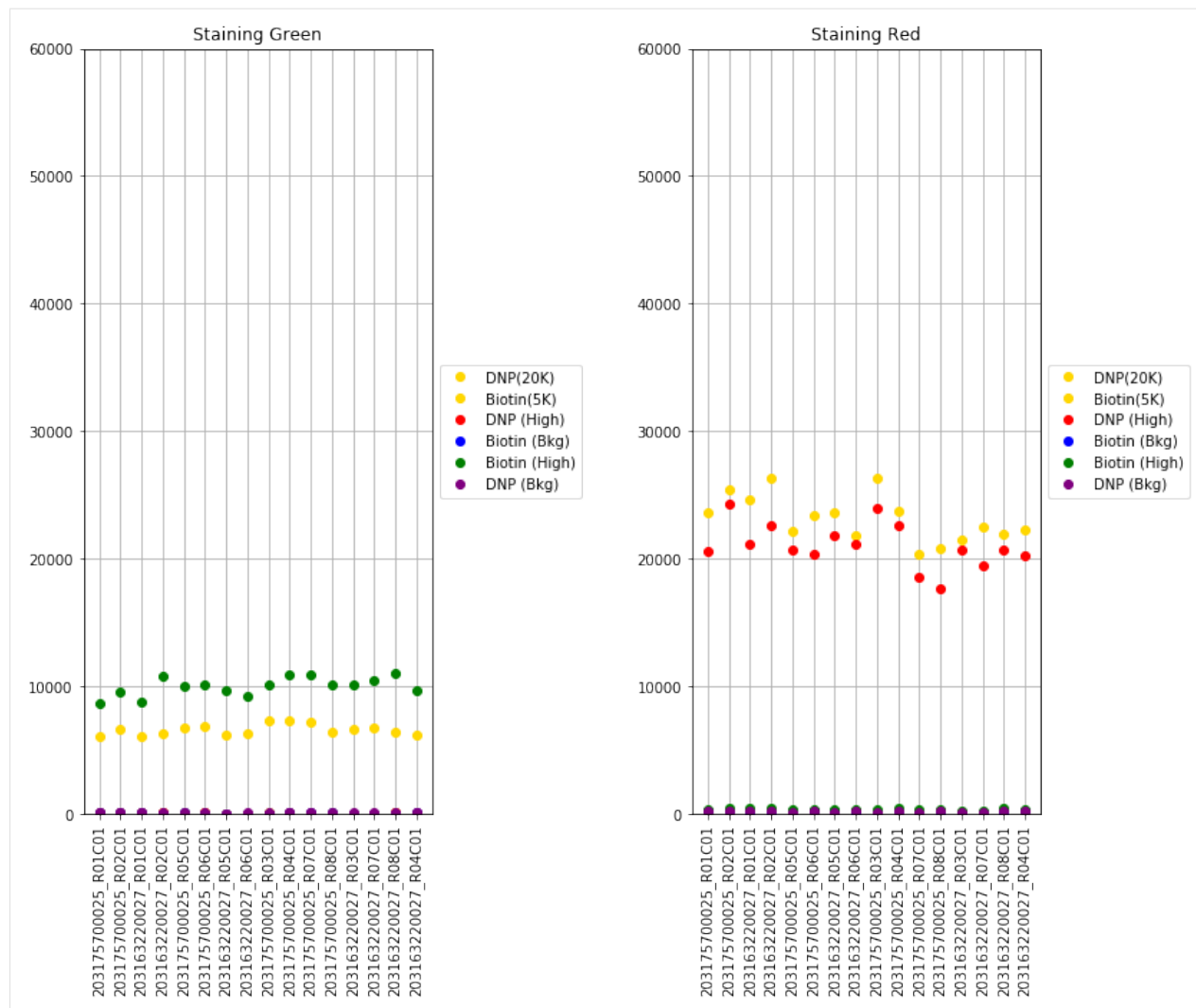
```
[4]: methylcheck.run_qc(filepath)
```

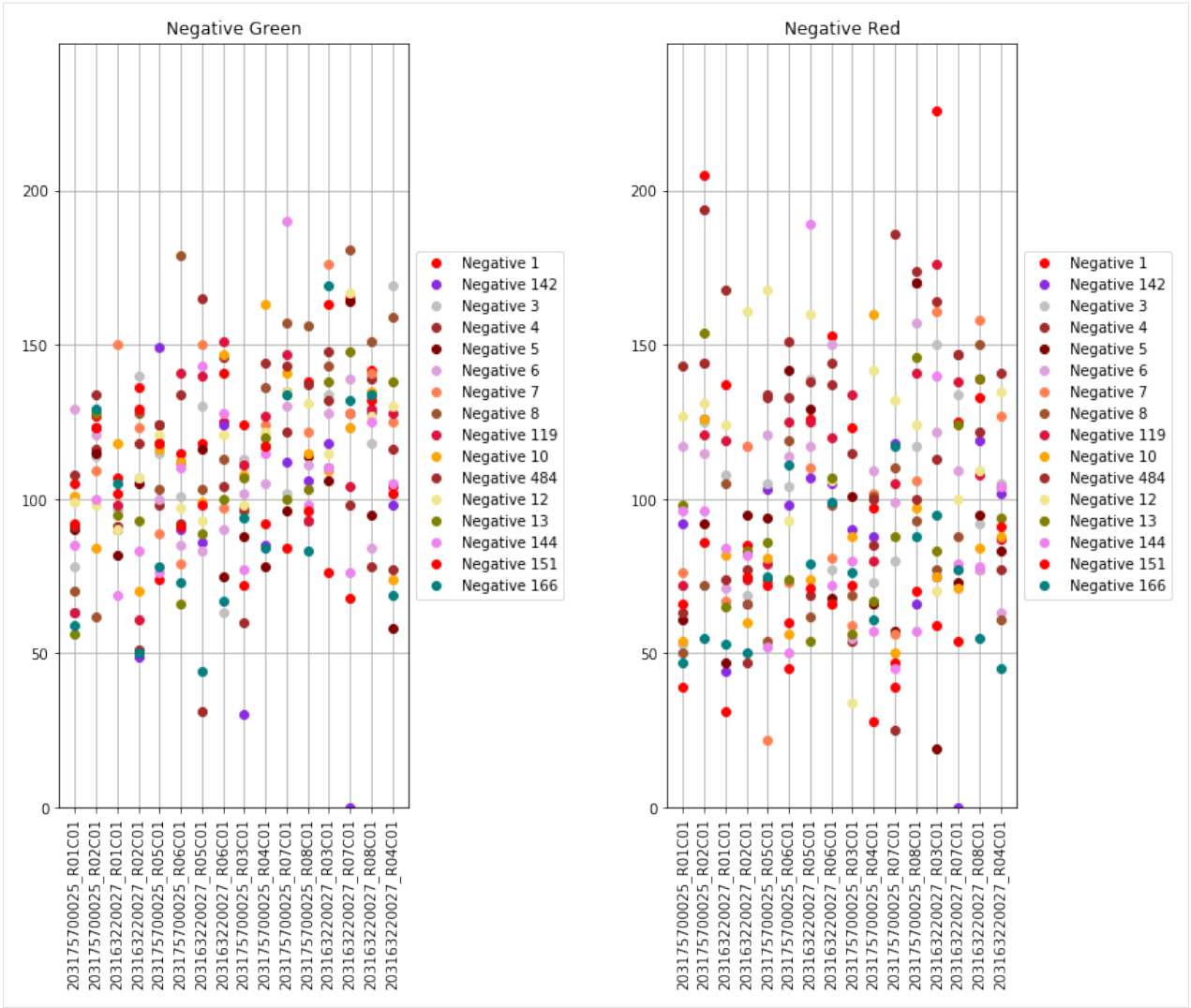


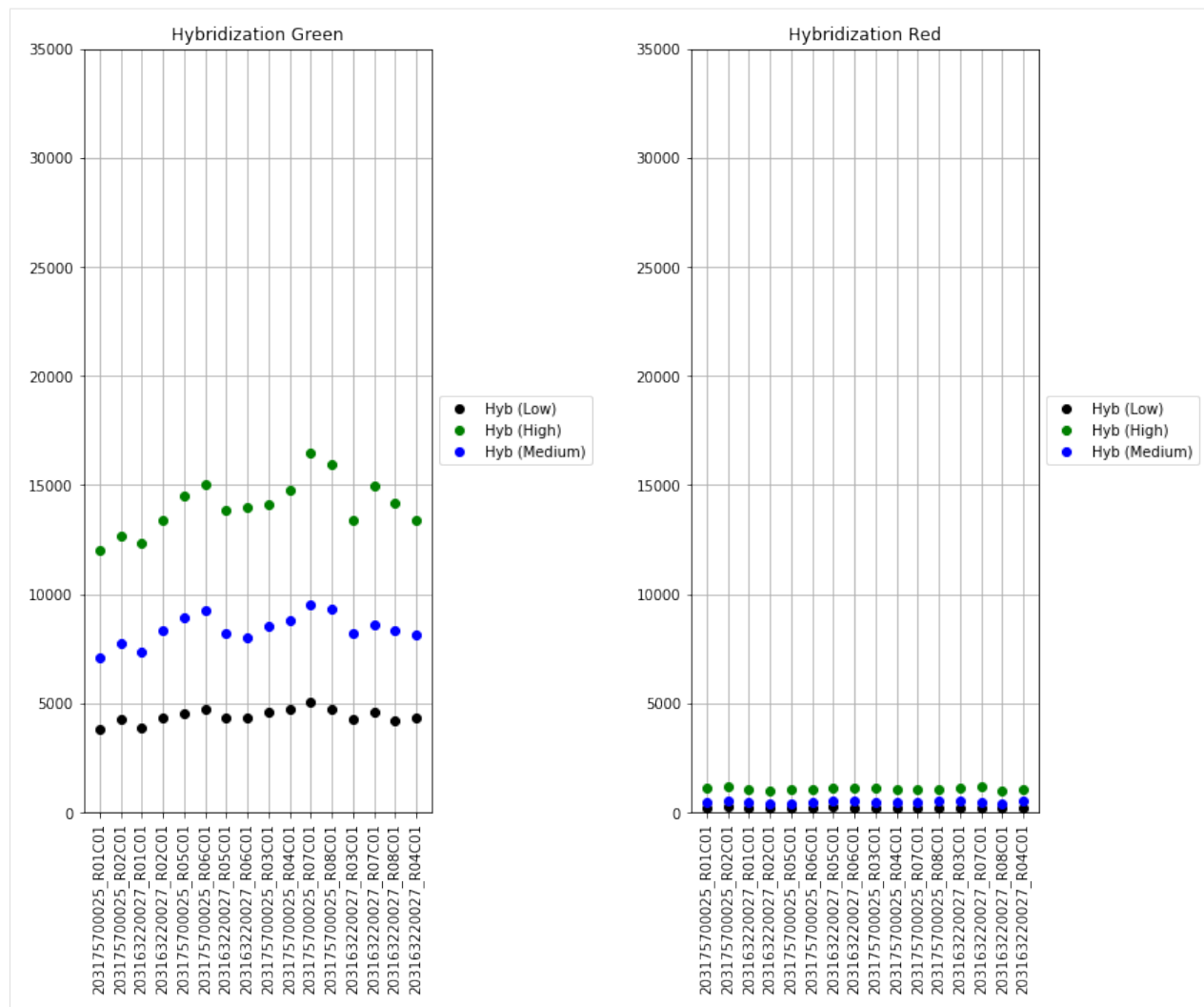


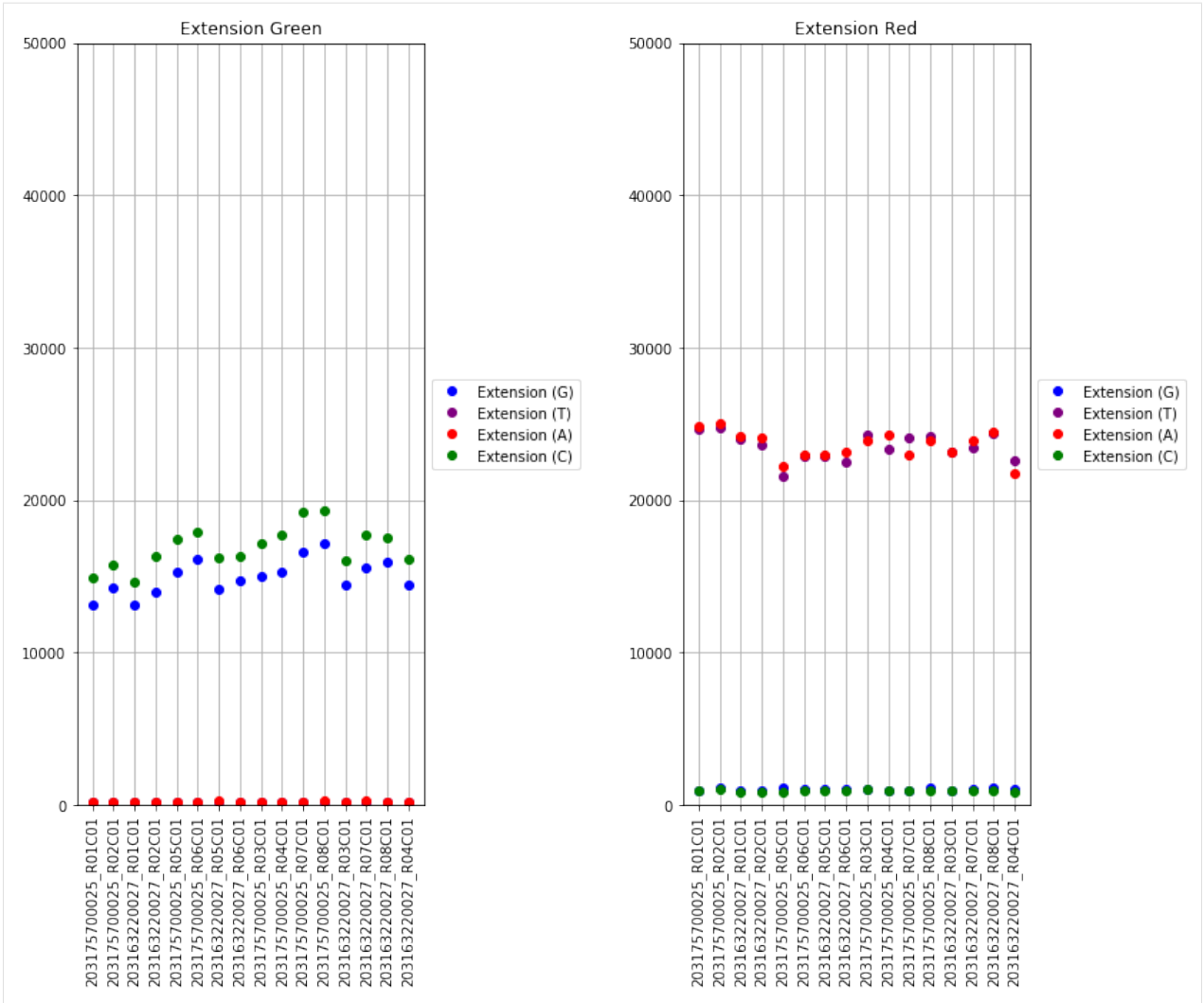
List of Bad Samples

```
[ '203163220027_R01C01', '203163220027_R03C01', '203163220027_R04C01', '203163220027_
↳ R05C01', '203163220027_R06C01', '203163220027_R08C01', '203175700025_R01C01',
↳ '203175700025_R02C01', '203175700025_R05C01', '203175700025_R06C01']
```

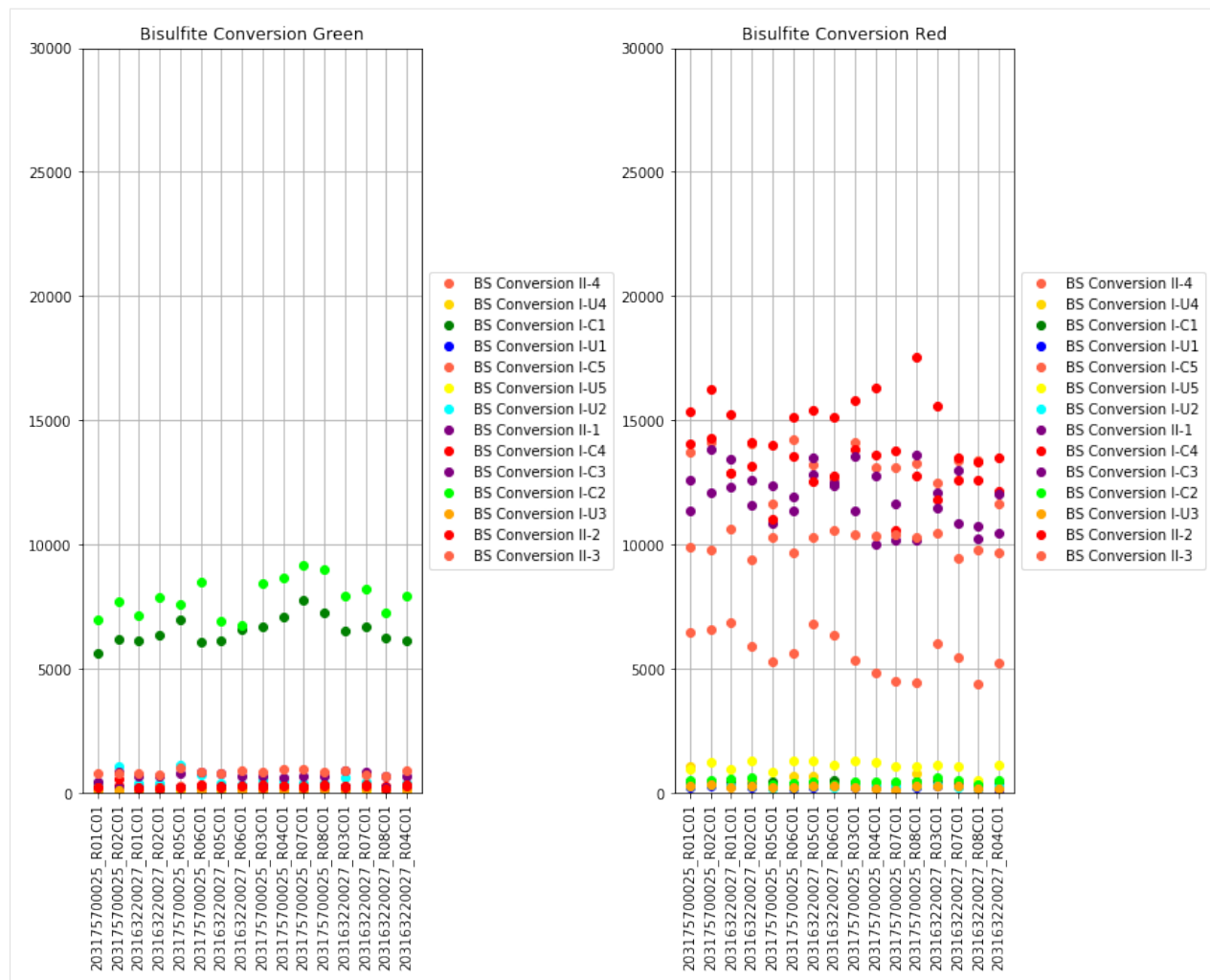


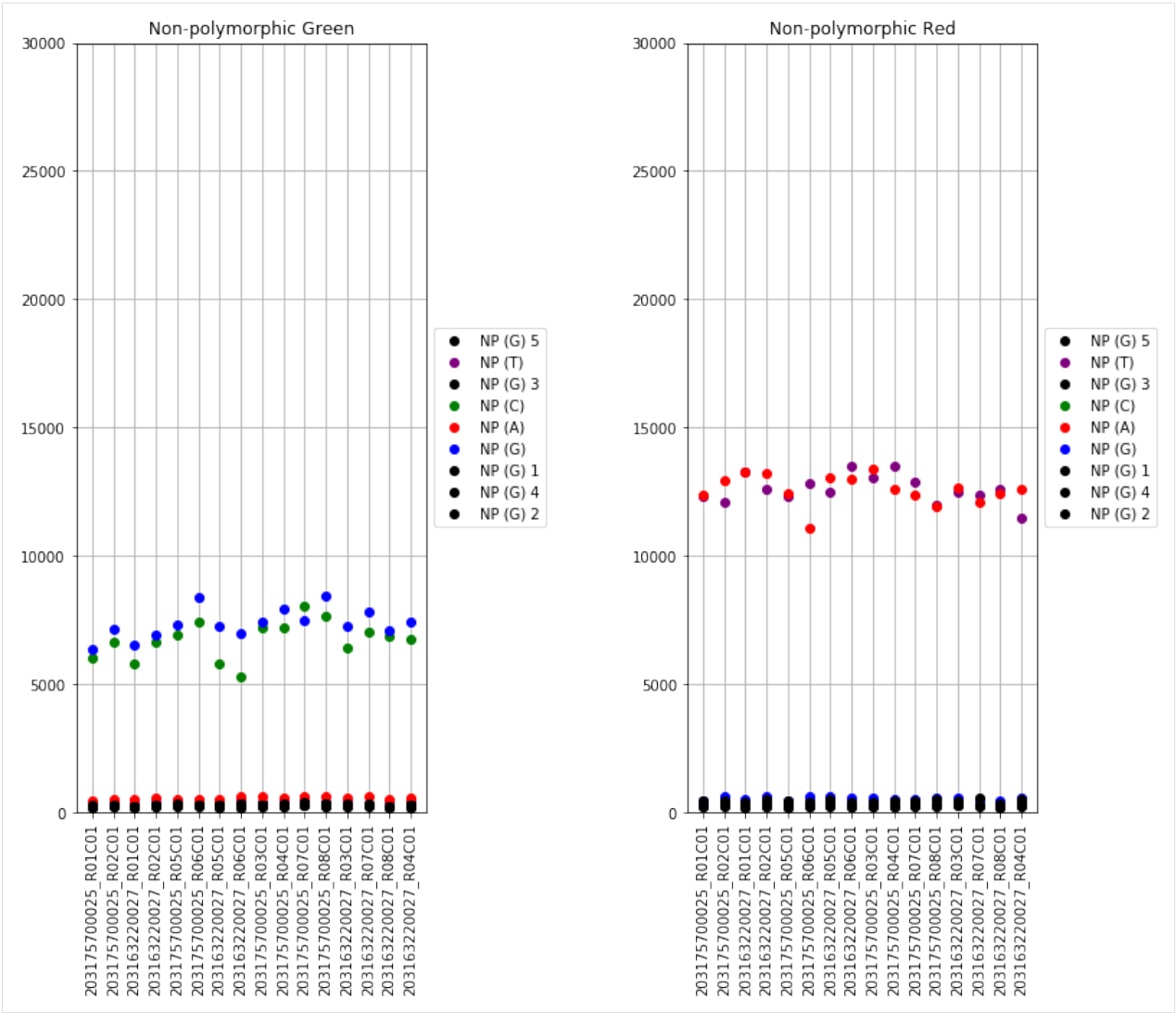


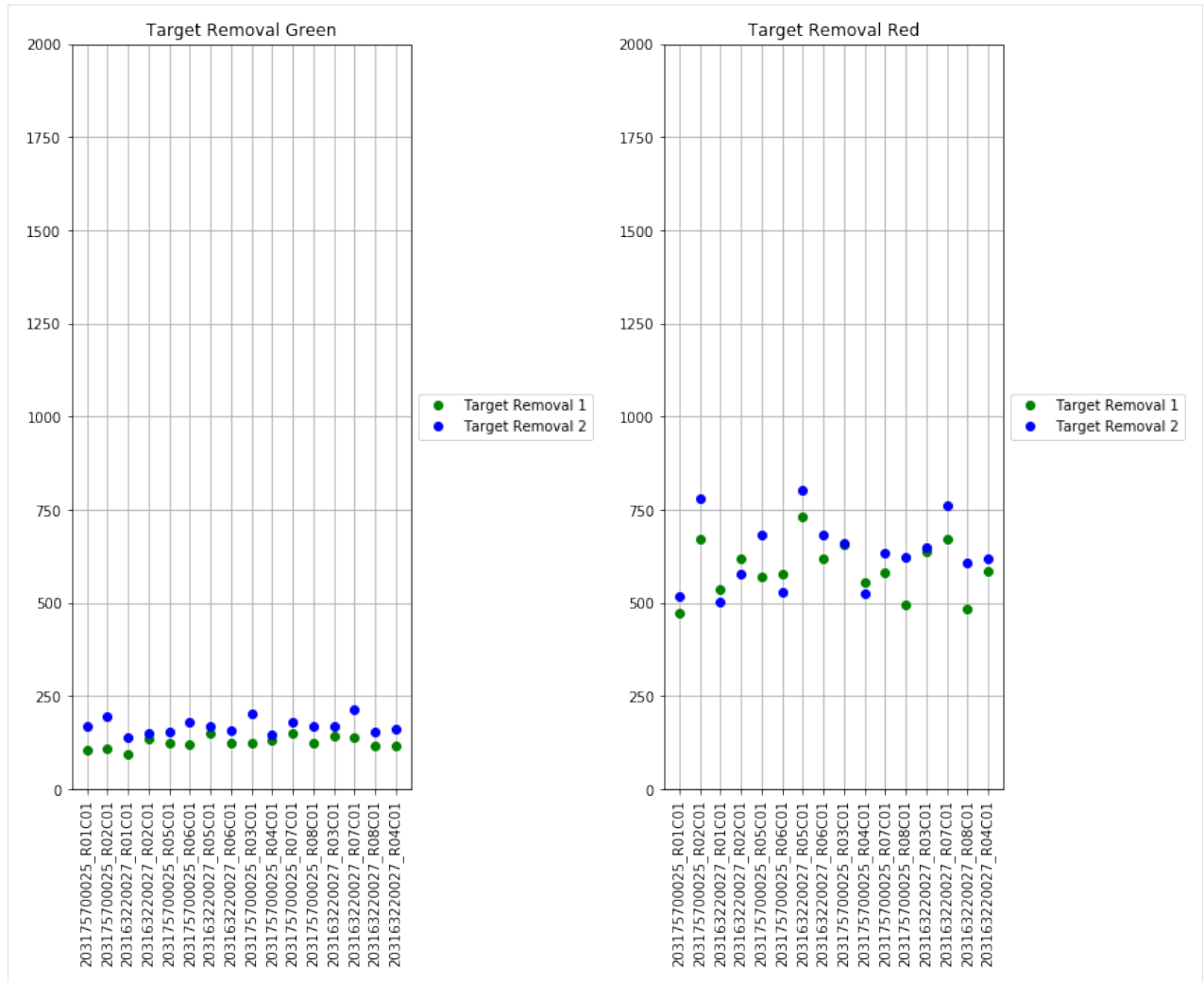


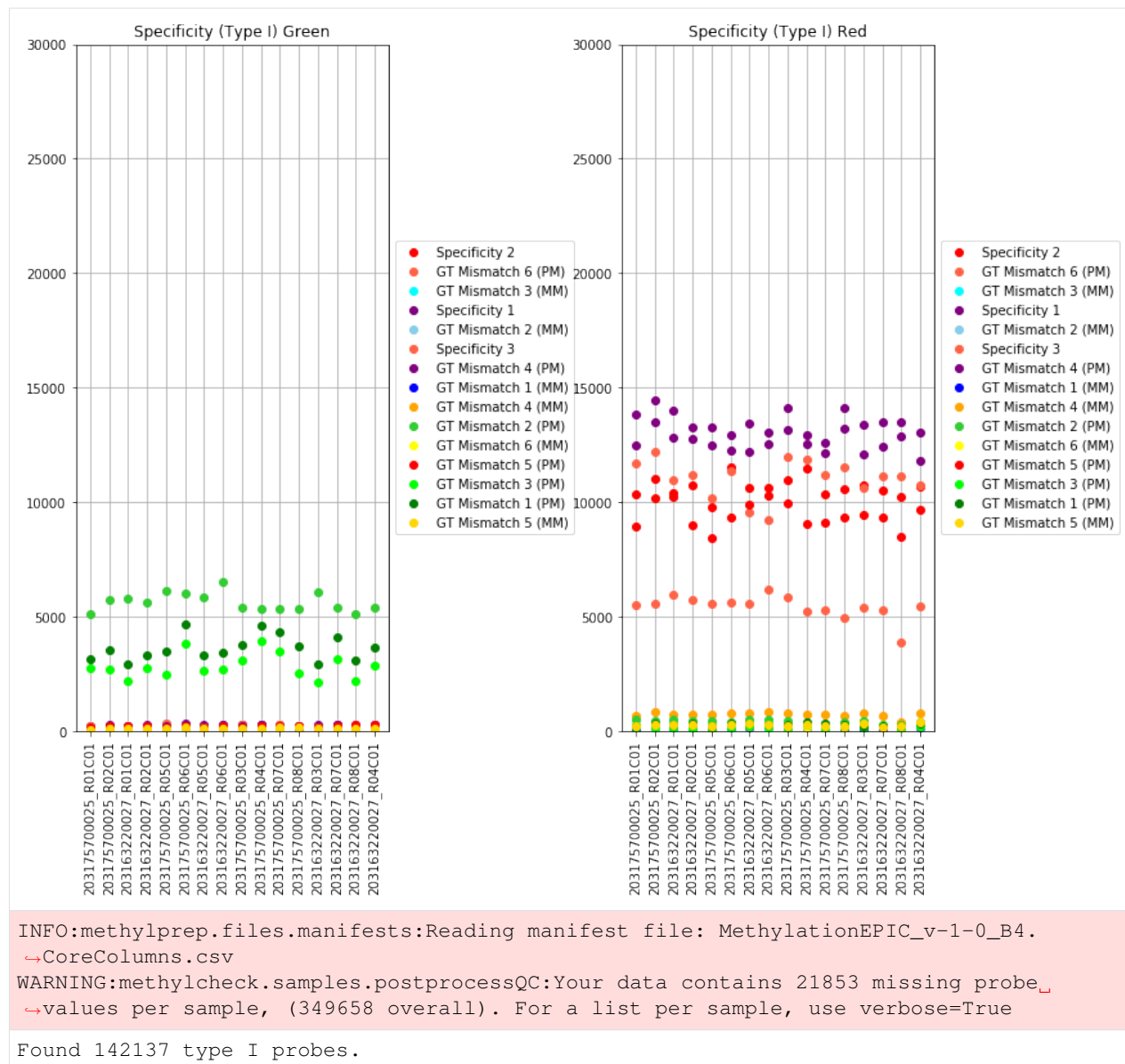


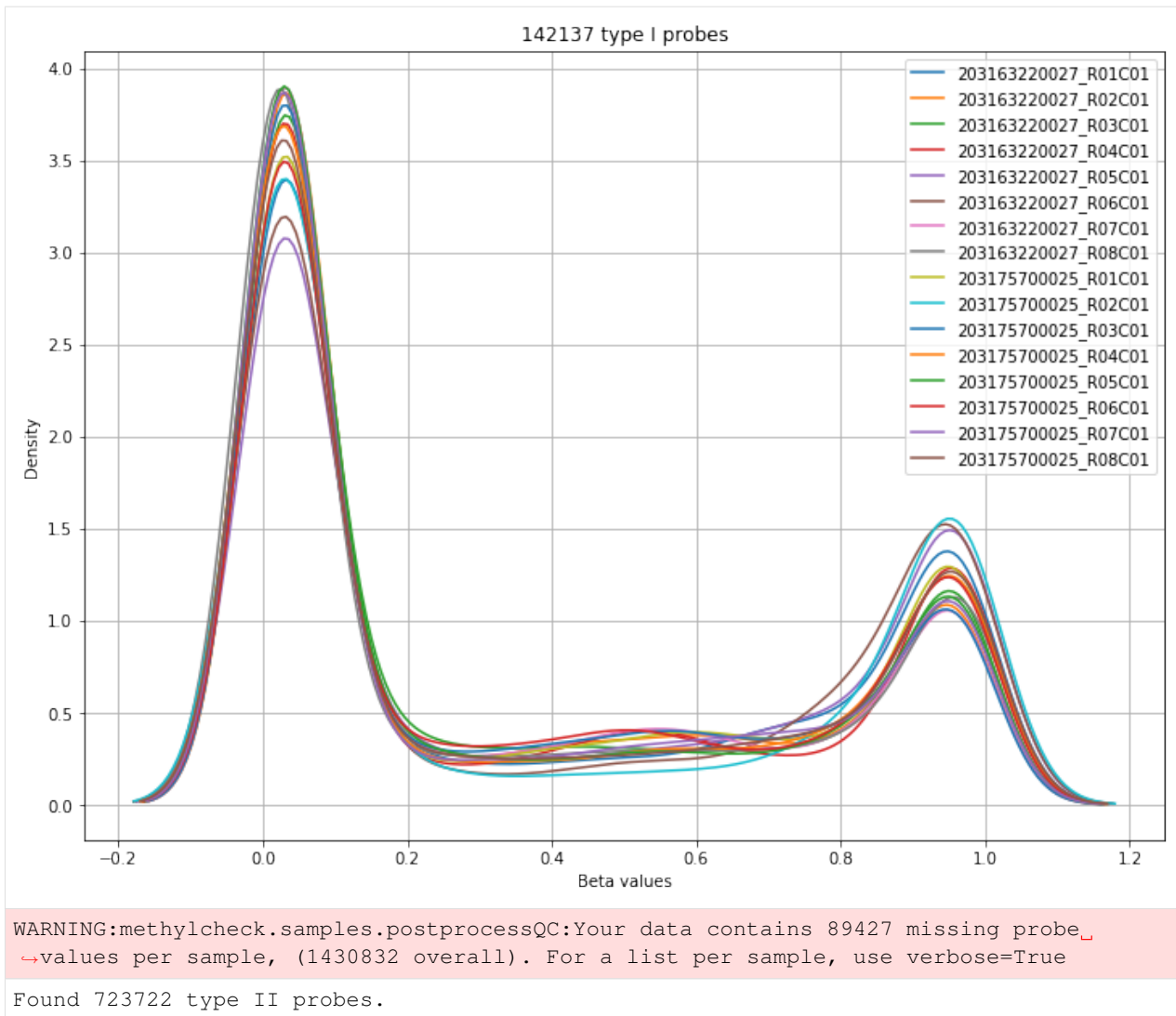


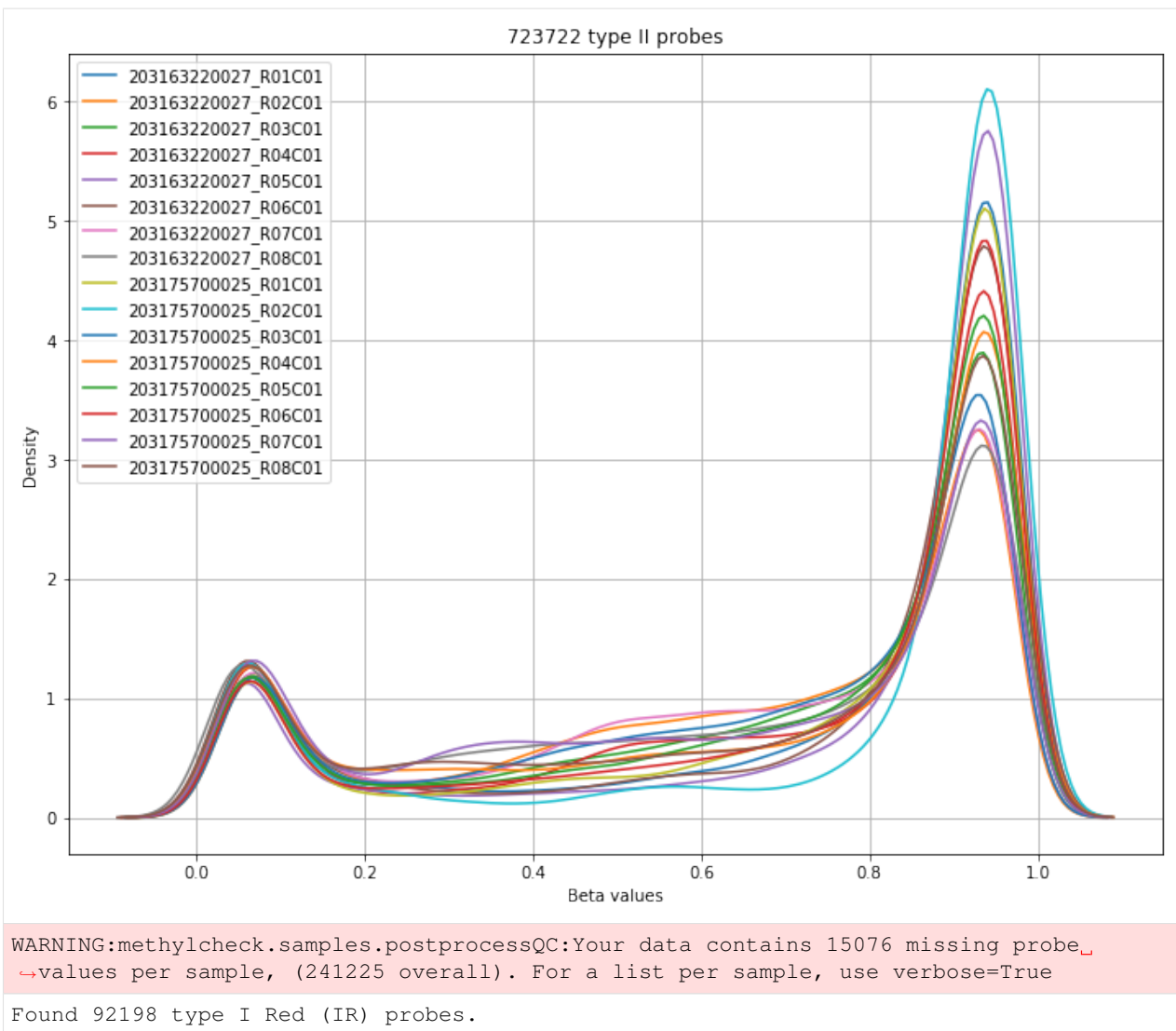


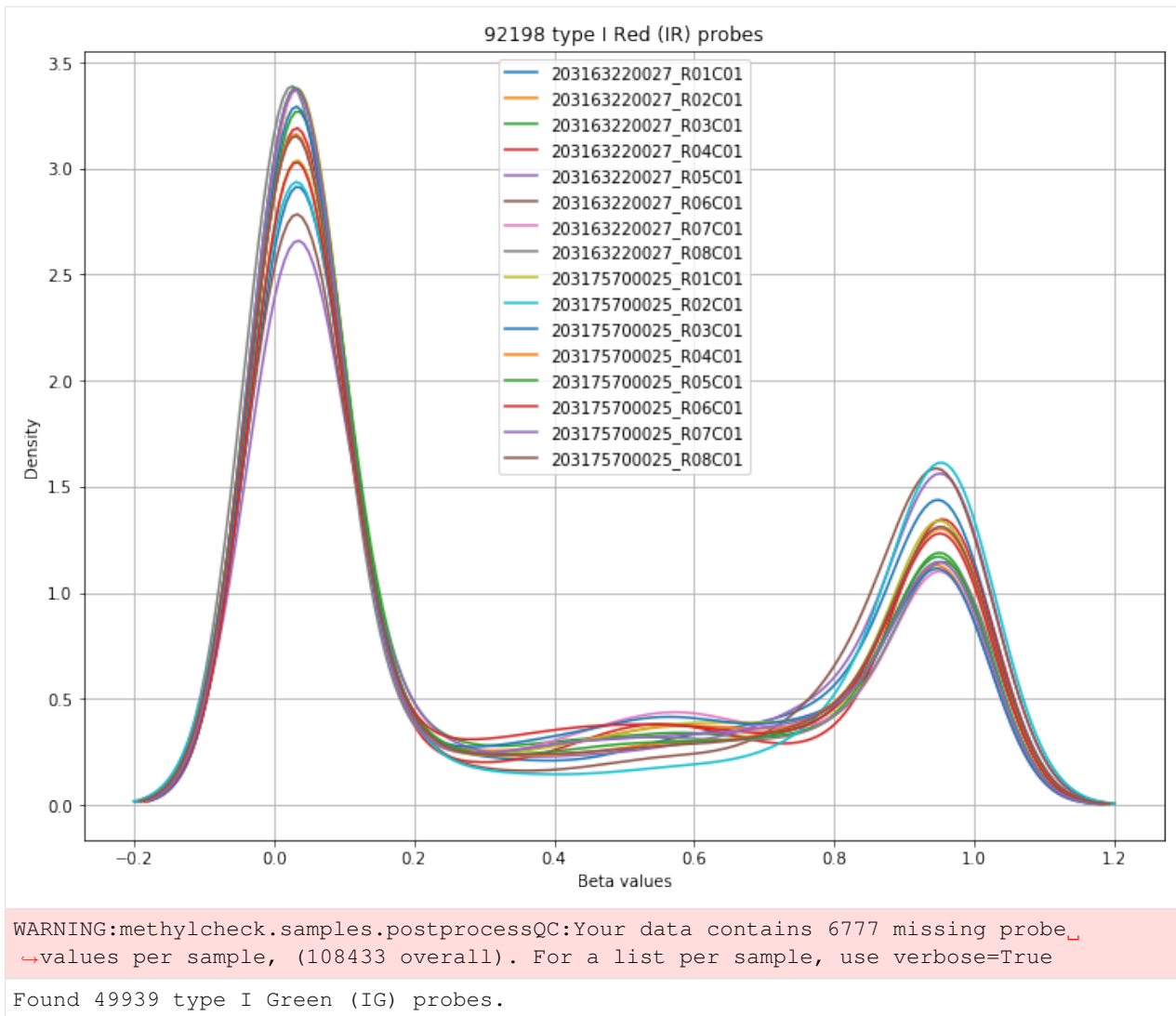


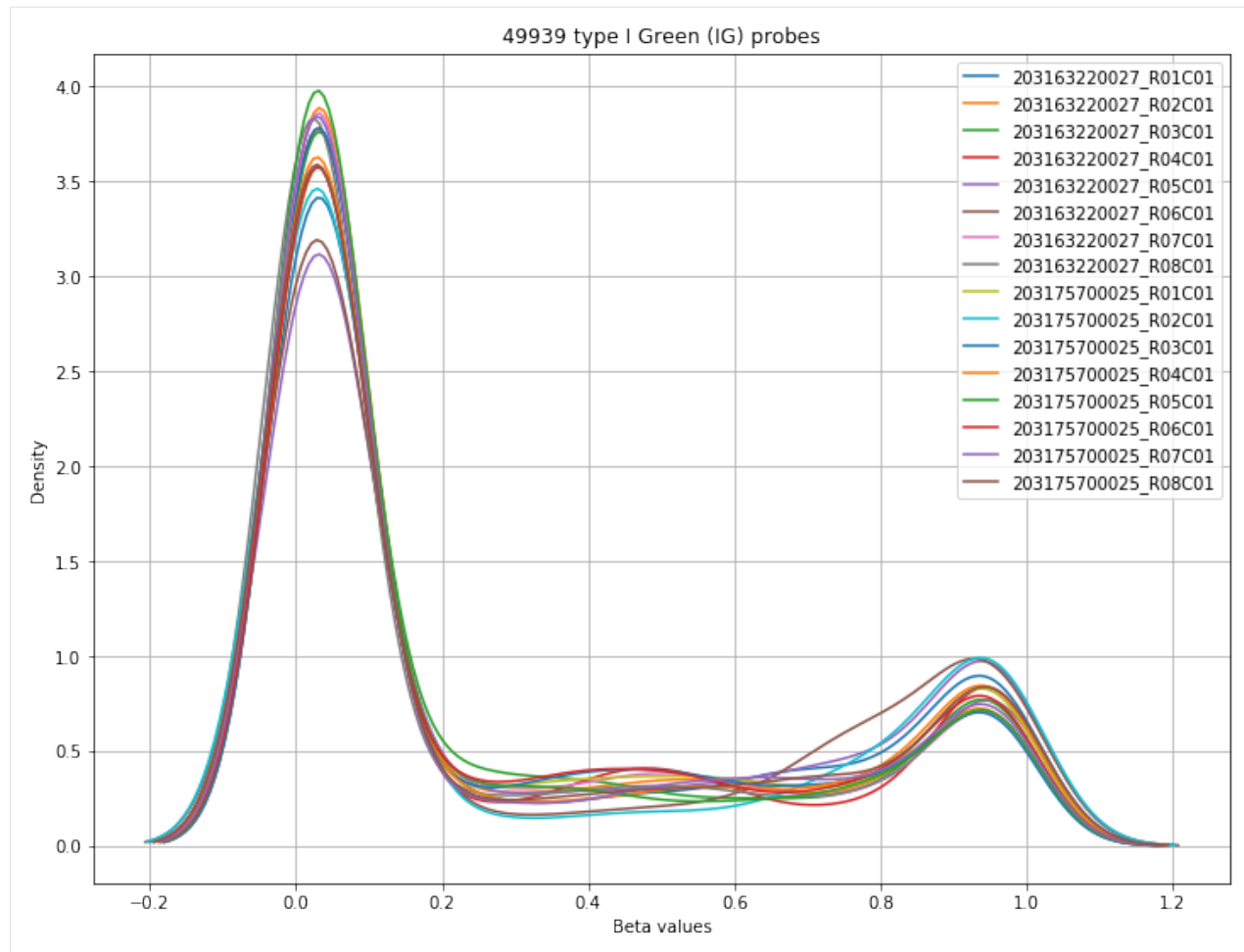












### 4.3.3 Predicting Sex

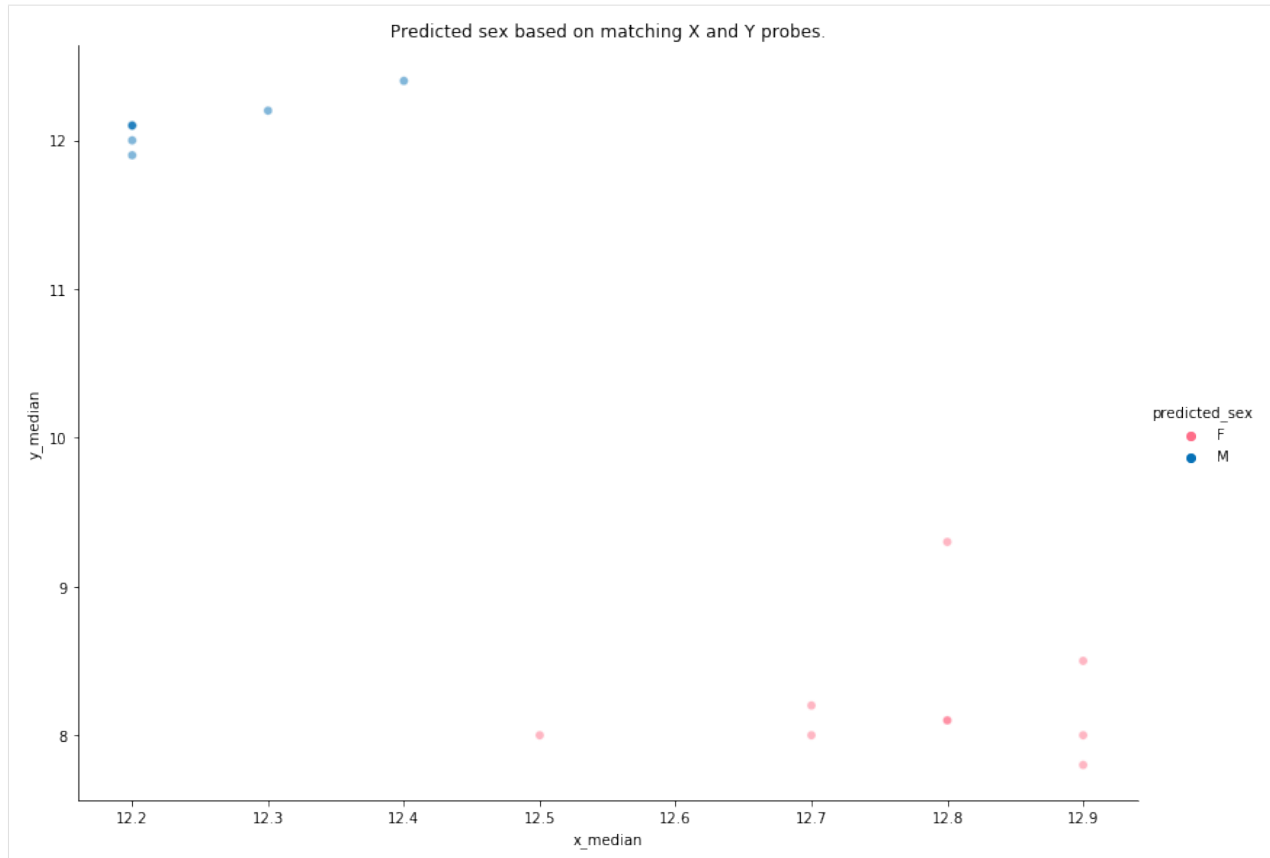
If you want to predict the sex of your samples without generating the entire QC report, you need the methylated and unmethylated .pkl files generated by methylprep and the `get_sex` function. See below for an example of its usage.

```
[5]: import methycheck
from pathlib import Path
filepath = Path('/Users/patriciagirardi/tutorial/GPL21145')
(meth, unmeth) = methycheck.load(filepath, format='meth_df')

methycheck.get_sex((meth, unmeth), plot=True)
```

```
100%| 16/16 [00:00<00:00, 491.79it/s]
100%| 16/16 [00:00<00:00, 1415.80it/s]
INFO:methycheck.load_processed:(865859, 16) (865859, 16)
INFO:methylprep.files.manifests:Reading manifest file: MethylationEPIC_v1-0_B4.
↪CoreColumns.csv
WARNING:methycheck.predict.sex:sample_failure_percent index did not align with_
↪output data index
```





```
[5]:
      x_median  y_median predicted_sex
203163220027_R01C01      12.7      8.0          F
203163220027_R02C01      12.3     12.2          M
203163220027_R03C01      12.8      9.3          F
203163220027_R04C01      12.8      8.1          F
203163220027_R05C01      12.2     12.0          M
203163220027_R06C01      12.2     12.1          M
203163220027_R07C01      12.9      8.5          F
203163220027_R08C01      12.7      8.2          F
203175700025_R01C01      12.5      8.0          F
203175700025_R02C01      12.2     12.1          M
203175700025_R03C01      12.9      8.0          F
203175700025_R04C01      12.9      7.8          F
203175700025_R05C01      12.8      8.1          F
203175700025_R06C01      12.2     11.9          M
203175700025_R07C01      12.4     12.4          M
203175700025_R08C01      12.2     12.1          M
```

Darker, smaller markers indicate less variability and a higher confidence in the predicted sex. Our data is pretty clean and has a good separation between the two sexes. It is occasionally the case that some samples will be misclassified. Those warrant further investigation as they are often poor quality samples that should be excluded.

#### 4.3.4 Report PDF tool

This is a fully customizable class template for making more advanced quality control reports. It will call a batch of plotting functions and compile a PDF with annotation and save file to disk.

- **kwargs:** filename, poobah\_max\_percent, pval\_cutoff, title, author, subject, keywords, outpath, path

If you want a quick one-line version with no further report editing, try `methylcheck.ReportPDF(runme=True)`. Otherwise, these are the 3 steps for using the `ReportPDF` class:

```
[8]: my_report = methylcheck.ReportPDF(filename='methylcheck_report.pdf',
                                         title="Example Report",
                                         path=filepath)

[9]: my_report.run_qc()

INFO:methylcheck.reports.qc_report:Data loaded
INFO:methylcheck.reports.qc_report:Beta Density Plot

['203163220027_R01C01', '203163220027_R02C01', '203163220027_R03C01', '203163220027_
↪R05C01', '203163220027_R07C01', '203163220027_R08C01', '203175700025_R01C01',
↪'203175700025_R03C01', '203175700025_R04C01', '203175700025_R05C01', '203175700025_
↪R06C01']

INFO:methylprep.files.manifests:Reading manifest file: MethylationEPIC_v-1-0_B4.
↪CoreColumns.csv
INFO:methylcheck.reports.qc_report:Beta MDS Plot
INFO:methylcheck.reports.qc_report:QC signal intensity plot
WARNING:methylcheck.qc_plot:Your poobah_values.pkl file contains missing values;
↪color coding will be inaccurate.
INFO:methylcheck.reports.qc_report:Control probes
INFO:methylcheck.reports.qc_report:Betas by probe type
INFO:methylprep.files.manifests:Reading manifest file: MethylationEPIC_v-1-0_B4.
↪CoreColumns.csv

[ ]: # Once everything has been added, you need to close the PDF before you can read it on
↪disk.
my_report.pdf.close()
# look in the path folder for a PDF file called "some_file_name.pdf" -- it will be
↪similar to the charts
# returned by methylprep.run_qc()
```

You can customize your `ReportPDF` and even pass in customized tables like this:

```
# generate report
report = methylcheck.qc_report.ReportPDF(
    path=working.name,
    poobah_max_percent=10,
    pval_cutoff=0.01,
    title='QC Report',
    author='FOXO Biosciences',
    subject="QC Report",
    keywords="methylation array",
    outpath=working.name,
    filename=report_filename,
    poobah=True,
    on_lambda=True,
    custom_tables=custom_tables,
    debug=True,
    order=['beta_density_plot', 'mds', 'auto_qc',
           'M_vs_U', 'qc_signal_intensity', 'controls',
           'probe_types'],
)
```

(continues on next page)

(continued from previous page)

```
report.run_qc()
report.pdf.close()
```

**Notes:**

- ‘on\_lambda’: if you are running this within an AWS lambda function in the cloud, the default paths of your manifest and other files will change. When True, on\_lambda will allow you to specify and override paths to your input, output, and manifest files.
- ‘path’: where to read files from.
- ‘outpath’: in this example, working.name is a python tempdir folder in a lambda virtual environment. Everything is processed there but saved by moving to an S3 bucket.
- ‘poobah’: whether the QC should run on samples that have failed probes removed (recommended)
- custom\_tables: passing in additional tables to the report. See the function’s help for more details (e.g. help(methyIcheck.qc\_report.ReportPDF)).
- ‘order’: Option to specify the order of charts in the REPORT.

## 4.4 Custom QC with pOOBAH Vales

This tutorial is meant for those who want to have more customization to their quality control of beta values. MethyIprep provides some automatic QC by default, but in this tutorial, we will go over how to do this manually, and with customizable parameters.

```
[1]: import methyIcheck
import pandas as pd
import numpy as np
```

Filepath of the processed files (Download and processing performed with MethyIprep package)

```
[2]: fpath = 'data/GPL13534/'
```

### 4.4.1 Load the Beta Values in a dataframe

The columns are each probe in the methylation array and the rows are each sample in the dataset. Note that if you want the dataframe in this orientation, you will need to transpose it.

The reason behind why we are using the format='beta\_csv' in methyIcheck.load is because this loads the raw beta values without any processing. By default, methyIprep does some QC on the beta values automatically and saves those new beta values in beta\_values.pkl. Specifically, it removes failed probes using Sesame pOOBAH method where a specific probe is classified as failed when the p-value  $\geq 0.05$ .

If you want to use the pOOBAH to mask beta values yourself, you must specify no\_poobah=True. Otherwise, it will mask them automatically when the CSV is loaded into a dataframe.

```
[22]: betas = methyIcheck.load('data/GPL13534', format='beta_csv', no_poobah=True).T
#betas.index.name = 'Samples'
print(betas.shape)
betas.head()
```

```
Files: 100%| 121/121 [00:52<00:00, 2.30it/s]
INFO:methyIcheck.load_processed:merging...
100%| 121/121 [00:00<00:00, 692.51it/s]
```

```
(121, 485577)
```

```
[22]: IlmnID          cg000000029  cg000000108  cg000000109  cg000000165  cg000000236  \
9996247040_R03C02      0.796      0.961      0.853      0.246      0.902
9996247040_R03C01      0.887      0.960      0.801      0.271      0.902
3998909005_R06C01      0.847      0.972      0.914      0.187      0.950
3998909005_R06C02      0.900      0.966      0.909      0.232      0.922
3998909206_R01C02      0.885      0.957      0.911      0.152      0.922

IlmnID          cg000000289  cg000000292  cg000000321  cg000000363  cg000000622  \
9996247040_R03C02      0.583      0.930      0.465      0.398      0.008
9996247040_R03C01      0.672      0.953      0.341      0.552      0.015
3998909005_R06C01      0.820      0.900      0.345      0.375      0.014
3998909005_R06C02      0.749      0.943      0.326      0.397      0.014
3998909206_R01C02      0.797      0.926      0.391      0.404      0.016

IlmnID          ...  rs7746156  rs798149  rs845016  rs877309  rs9292570  \
9996247040_R03C02  ...      0.468      0.374      0.079      0.016      0.978
9996247040_R03C01  ...      0.971      0.396      0.059      0.444      0.967
3998909005_R06C01  ...      0.512      0.018      0.469      0.539      0.019
3998909005_R06C02  ...      0.508      0.984      0.922      0.024      0.019
3998909206_R01C02  ...      0.045      0.977      0.491      0.531      0.480

IlmnID          rs9363764  rs939290  rs951295  rs966367  rs9839873
9996247040_R03C02      0.544      0.961      0.981      0.883      0.614
9996247040_R03C01      0.042      0.537      0.967      0.582      0.333
3998909005_R06C01      0.951      0.551      0.969      0.959      0.950
3998909005_R06C02      0.962      0.582      0.536      0.945      0.944
3998909206_R01C02      0.075      0.969      0.535      0.946      0.162
```

```
[5 rows x 485577 columns]
```

When loading the betas from the CSV, there are still control probes in your resulting dataframe. The cell below shows how to remove all of the control probes from you betas dataframe.

```
[23]: rs_probes = betas.columns[betas.columns.str.startswith('rs')]
betas_nocontrol = betas.drop(rs_probes, axis=1)
print(betas_nocontrol.shape)
betas_nocontrol = betas_nocontrol.T[betas_nocontrol.index.sort_values()].T
betas_nocontrol
```

```
(121, 485512)
```

```
[23]: IlmnID          cg000000029  cg000000108  cg000000109  cg000000165  \
100946230055_R04C01      0.864      0.971      0.925      0.288
100946230056_R04C01      0.854      0.978      0.930      0.215
100946230056_R04C02      0.879      0.958      0.866      0.257
101032570143_R04C02      0.837      0.968      0.911      0.334
101032570152_R04C01      0.813      0.971      0.928      0.164
...
9996247054_R03C01      0.840      0.957      0.871      0.253
9996247054_R03C02      0.864      0.963      0.864      0.194
9996247055_R03C01      0.817      0.956      0.842      0.292
9996247055_R03C02      0.801      0.965      0.869      0.357
9996247056_R05C02      0.837      0.976      0.945      0.244
```

(continues on next page)

(continued from previous page)

IlmnID	cg00000236	cg00000289	cg00000292	cg00000321	\
100946230055_R04C01	0.935	0.654	0.945	0.378	
100946230056_R04C01	0.932	0.639	0.971	0.421	
100946230056_R04C02	0.899	0.604	0.971	0.191	
101032570143_R04C02	0.918	0.765	0.951	0.435	
101032570152_R04C01	0.934	0.810	0.955	0.358	
...	...	...	...	...	
9996247054_R03C01	0.917	0.673	0.932	0.374	
9996247054_R03C02	0.885	0.682	0.929	0.393	
9996247055_R03C01	0.887	0.671	0.903	0.423	
9996247055_R03C02	0.893	0.647	0.962	0.340	
9996247056_R05C02	0.957	0.759	0.927	0.402	
IlmnID	cg00000363	cg00000622	...	ch.X.93511680F	\
100946230055_R04C01	0.468	0.010	...	0.046	
100946230056_R04C01	0.397	0.012	...	0.034	
100946230056_R04C02	0.560	0.012	...	0.038	
101032570143_R04C02	0.456	0.011	...	0.034	
101032570152_R04C01	0.372	0.011	...	0.044	
...	...	...	...	...	
9996247054_R03C01	0.374	0.012	...	0.050	
9996247054_R03C02	0.418	0.019	...	0.047	
9996247055_R03C01	0.490	0.014	...	0.037	
9996247055_R03C02	0.467	0.013	...	0.046	
9996247056_R05C02	0.454	0.013	...	0.049	
IlmnID	ch.X.938089F	ch.X.94051109R	ch.X.94260649R	\	
100946230055_R04C01	0.036	0.031	0.171		
100946230056_R04C01	0.040	0.034	0.105		
100946230056_R04C02	0.057	0.045	0.253		
101032570143_R04C02	0.098	0.058	0.329		
101032570152_R04C01	0.058	0.036	0.171		
...	...	...	...		
9996247054_R03C01	0.037	0.040	0.234		
9996247054_R03C02	0.048	0.029	0.155		
9996247055_R03C01	0.042	0.026	0.191		
9996247055_R03C02	0.045	0.025	0.157		
9996247056_R05C02	0.030	0.025	0.052		
IlmnID	ch.X.967194F	ch.X.97129969R	ch.X.97133160R	\	
100946230055_R04C01	0.150	0.108	0.076		
100946230056_R04C01	0.149	0.094	0.059		
100946230056_R04C02	0.232	0.186	0.090		
101032570143_R04C02	0.392	0.447	0.206		
101032570152_R04C01	0.350	0.222	0.137		
...	...	...	...		
9996247054_R03C01	0.355	0.177	0.052		
9996247054_R03C02	0.283	0.187	0.065		
9996247055_R03C01	0.395	0.248	0.057		
9996247055_R03C02	0.384	0.143	0.059		
9996247056_R05C02	0.123	0.046	0.045		
IlmnID	ch.X.97651759F	ch.X.97737721F	ch.X.98007042R		
100946230055_R04C01	0.022	0.063	0.078		
100946230056_R04C01	0.019	0.058	0.057		
100946230056_R04C02	0.022	0.073	0.128		

(continues on next page)

(continued from previous page)

```

101032570143_R04C02      0.032      0.070      0.142
101032570152_R04C01      0.031      0.061      0.152
...
9996247054_R03C01      0.029      0.055      0.099
9996247054_R03C02      0.037      0.086      0.180
9996247055_R03C01      0.031      0.067      0.132
9996247055_R03C02      0.024      0.055      0.097
9996247056_R05C02      0.026      0.065      0.063

[121 rows x 485512 columns]

```

## 4.4.2 Load p-values in a dataframe

This is reading in the pOOBAH values to a dataframe, and should have the same dimensions as the betas dataframe. Each cell in this dataframe is a p-value for each probe for a specific sample. If a p-value is  $\geq 0.05$ , then it's more likely that that specific probe for that sample failed. A failed probe means that the true probes signal is not distinguishable from the background fluorescence.

```

[19]: p = pd.read_pickle('data/GPL13534/poobah_values.pkl').T
      #p.index.name = 'Samples'
      print(p.shape)
      assert p.shape == betas_nocontrol.shape
      print(f'Number of p-values >= 0.05: {(p>=0.05).sum().sum()}')
      p = p.T[p.index.sort_values()].T
      p.head()

```

```

(121, 485512)
Number of p-values >= 0.05: 1546688

```

```

[19]: IlmnID      cg00000029  cg000000108  cg000000109  cg000000165  \
100946230055_R04C01      0.003      0.000      0.002      0.038
100946230056_R04C01      0.004      0.000      0.002      0.026
100946230056_R04C02      0.018      0.001      0.028      0.096
101032570143_R04C02      0.004      0.001      0.004      0.054
101032570152_R04C01      0.002      0.000      0.002      0.023

IlmnID      cg000000236  cg000000289  cg000000292  cg000000321  \
100946230055_R04C01      0.002      0.057      0.0      0.002
100946230056_R04C01      0.001      0.082      0.0      0.001
100946230056_R04C02      0.007      0.130      0.0      0.005
101032570143_R04C02      0.002      0.045      0.0      0.001
101032570152_R04C01      0.001      0.018      0.0      0.002

IlmnID      cg000000363  cg000000622  ...  ch.X.93511680F  \
100946230055_R04C01      0.001      0.0      ...      NaN
100946230056_R04C01      0.001      0.0      ...      NaN
100946230056_R04C02      0.002      0.0      ...      NaN
101032570143_R04C02      0.001      0.0      ...      NaN
101032570152_R04C01      0.001      0.0      ...      NaN

IlmnID      ch.X.938089F  ch.X.94051109R  ch.X.94260649R  \
100946230055_R04C01      0.006      0.004      0.073
100946230056_R04C01      0.004      0.003      0.038
100946230056_R04C02      0.010      0.006      0.127
101032570143_R04C02      0.025      0.014      0.241
101032570152_R04C01      0.014      0.006      0.086

```

(continues on next page)

(continued from previous page)

```

IlmnID          ch.X.967194F  ch.X.97129969R  ch.X.97133160R  \
100946230055_R04C01          NaN          0.037          NaN
100946230056_R04C01          NaN          0.035          NaN
100946230056_R04C02          NaN          0.082          NaN
101032570143_R04C02          NaN          0.535          NaN
101032570152_R04C01          NaN          0.117          NaN

IlmnID          ch.X.97651759F  ch.X.97737721F  ch.X.98007042R
100946230055_R04C01          0.001          NaN          NaN
100946230056_R04C01          0.001          NaN          NaN
100946230056_R04C02          0.001          NaN          NaN
101032570143_R04C02          0.003          NaN          NaN
101032570152_R04C01          0.003          NaN          NaN

```

[5 rows x 485512 columns]

### 4.4.3 Mask Beta values where probe fails

When the p-value of a probe for a specific sample  $\geq 0.05$ , it is more likely that the probe has failed, which means that the beta value for that probe may not be accurate. Because of this, it is a good idea to mask these beta values with a NULL value.

```

[24]: cutoff = 0.05
      betas_filtered = betas_nocontrol.mask((p>=cutoff), np.nan)

      print(betas_filtered.shape)
      print(f'Masked {betas_filtered.isna().sum().sum() - betas_nocontrol.isna().sum().sum()} beta values')
      betas_filtered

```

(121, 485512)

Masked 1546688 beta values

```

[24]: IlmnID          cg00000029  cg000000108  cg000000109  cg000000165  \
100946230055_R04C01          0.864          0.971          0.925          0.288
100946230056_R04C01          0.854          0.978          0.930          0.215
100946230056_R04C02          0.879          0.958          0.866          NaN
101032570143_R04C02          0.837          0.968          0.911          NaN
101032570152_R04C01          0.813          0.971          0.928          0.164
...
9996247054_R03C01          0.840          0.957          0.871          0.253
9996247054_R03C02          0.864          0.963          0.864          0.194
9996247055_R03C01          0.817          0.956          0.842          NaN
9996247055_R03C02          0.801          0.965          0.869          NaN
9996247056_R05C02          0.837          0.976          0.945          0.244

IlmnID          cg000000236  cg000000289  cg000000292  cg000000321  \
100946230055_R04C01          0.935          NaN          0.945          0.378
100946230056_R04C01          0.932          NaN          0.971          0.421
100946230056_R04C02          0.899          NaN          0.971          0.191
101032570143_R04C02          0.918          0.765          0.951          0.435
101032570152_R04C01          0.934          0.810          0.955          0.358
...
9996247054_R03C01          0.917          0.673          0.932          0.374
9996247054_R03C02          0.885          NaN          0.929          0.393

```

(continues on next page)

(continued from previous page)

9996247055_R03C01	0.887	0.671	0.903	0.423
9996247055_R03C02	0.893	NaN	0.962	0.340
9996247056_R05C02	0.957	0.759	0.927	0.402
IlmnID	cg00000363	cg00000622	...	ch.X.93511680F \
100946230055_R04C01	0.468	0.010	...	0.046
100946230056_R04C01	0.397	0.012	...	0.034
100946230056_R04C02	0.560	0.012	...	0.038
101032570143_R04C02	0.456	0.011	...	0.034
101032570152_R04C01	0.372	0.011	...	0.044
...	...	...	...	...
9996247054_R03C01	0.374	0.012	...	0.050
9996247054_R03C02	0.418	0.019	...	0.047
9996247055_R03C01	0.490	0.014	...	0.037
9996247055_R03C02	0.467	0.013	...	0.046
9996247056_R05C02	0.454	0.013	...	0.049
IlmnID	ch.X.938089F	ch.X.94051109R	ch.X.94260649R	\
100946230055_R04C01	0.036	0.031	NaN	
100946230056_R04C01	0.040	0.034	0.105	
100946230056_R04C02	0.057	0.045	NaN	
101032570143_R04C02	0.098	0.058	NaN	
101032570152_R04C01	0.058	0.036	NaN	
...	...	...	...	
9996247054_R03C01	0.037	0.040	NaN	
9996247054_R03C02	0.048	0.029	0.155	
9996247055_R03C01	0.042	0.026	NaN	
9996247055_R03C02	0.045	0.025	0.157	
9996247056_R05C02	0.030	0.025	0.052	
IlmnID	ch.X.967194F	ch.X.97129969R	ch.X.97133160R	\
100946230055_R04C01	0.150	0.108	0.076	
100946230056_R04C01	0.149	0.094	0.059	
100946230056_R04C02	0.232	NaN	0.090	
101032570143_R04C02	0.392	NaN	0.206	
101032570152_R04C01	0.350	NaN	0.137	
...	...	...	...	
9996247054_R03C01	0.355	NaN	0.052	
9996247054_R03C02	0.283	NaN	0.065	
9996247055_R03C01	0.395	NaN	0.057	
9996247055_R03C02	0.384	0.143	0.059	
9996247056_R05C02	0.123	0.046	0.045	
IlmnID	ch.X.97651759F	ch.X.97737721F	ch.X.98007042R	
100946230055_R04C01	0.022	0.063	0.078	
100946230056_R04C01	0.019	0.058	0.057	
100946230056_R04C02	0.022	0.073	0.128	
101032570143_R04C02	0.032	0.070	0.142	
101032570152_R04C01	0.031	0.061	0.152	
...	...	...	...	
9996247054_R03C01	0.029	0.055	0.099	
9996247054_R03C02	0.037	0.086	0.180	
9996247055_R03C01	0.031	0.067	0.132	
9996247055_R03C02	0.024	0.055	0.097	
9996247056_R05C02	0.026	0.065	0.063	
[121 rows x 485512 columns]				



Start here if you have already masked your beta values based on p-values or had that done automatically

#### 4.4.4 Remove Samples based on Percent or Number of Failed Probes

```
[40]: percent_cutoff = 0.2 #use a percent in decimal format (20% = 0.2)
qc_betas = betas_filtered[~((betas_filtered.T.isna().sum() / betas_filtered.shape[1])
    ↳ > 0.2)]

#if you want to remove samples based off a number threshold rather than a percentage,
    ↳ use the following 2 lines:
#number_cutoff = 20000
#qc_betas = betas_filtered[~(betas_filtered.T.isna().sum() >= number_cutoff)]

print(f'{betas_filtered.shape[0] - qc_betas.shape[0]} sample(s) removed because of
    ↳ pOOBAH failure')
print(f'Sample(s) removed: {set(betas_filtered.index) - set(qc_betas.index)}')
print(qc_betas.shape)
qc_betas
```

```
1 sample(s) removed because of poobah failure
Sample(s) removed: {'101032570169_R04C02'}
(120, 485512)
```

```
[40]: IlmnID          cg000000029  cg000000108  cg000000109  cg000000165  \
100946230055_R04C01      0.864      0.971      0.925      0.288
100946230056_R04C01      0.854      0.978      0.930      0.215
100946230056_R04C02      0.879      0.958      0.866      NaN
101032570143_R04C02      0.837      0.968      0.911      NaN
101032570152_R04C01      0.813      0.971      0.928      0.164
...
9996247054_R03C01      0.840      0.957      0.871      0.253
9996247054_R03C02      0.864      0.963      0.864      0.194
9996247055_R03C01      0.817      0.956      0.842      NaN
9996247055_R03C02      0.801      0.965      0.869      NaN
9996247056_R05C02      0.837      0.976      0.945      0.244

IlmnID          cg000000236  cg000000289  cg000000292  cg000000321  \
100946230055_R04C01      0.935      NaN      0.945      0.378
100946230056_R04C01      0.932      NaN      0.971      0.421
100946230056_R04C02      0.899      NaN      0.971      0.191
101032570143_R04C02      0.918      0.765      0.951      0.435
101032570152_R04C01      0.934      0.810      0.955      0.358
...
9996247054_R03C01      0.917      0.673      0.932      0.374
9996247054_R03C02      0.885      NaN      0.929      0.393
9996247055_R03C01      0.887      0.671      0.903      0.423
9996247055_R03C02      0.893      NaN      0.962      0.340
9996247056_R05C02      0.957      0.759      0.927      0.402

IlmnID          cg000000363  cg000000622  ...  ch.X.93511680F  \
100946230055_R04C01      0.468      0.010      ...      0.046
100946230056_R04C01      0.397      0.012      ...      0.034
100946230056_R04C02      0.560      0.012      ...      0.038
101032570143_R04C02      0.456      0.011      ...      0.034
101032570152_R04C01      0.372      0.011      ...      0.044
...
9996247054_R03C01      0.374      0.012      ...      0.050
```

(continues on next page)

(continued from previous page)

9996247054_R03C02	0.418	0.019	...	0.047
9996247055_R03C01	0.490	0.014	...	0.037
9996247055_R03C02	0.467	0.013	...	0.046
9996247056_R05C02	0.454	0.013	...	0.049
IlmnID	ch.X.938089F	ch.X.94051109R	ch.X.94260649R	\
100946230055_R04C01	0.036	0.031	NaN	
100946230056_R04C01	0.040	0.034	0.105	
100946230056_R04C02	0.057	0.045	NaN	
101032570143_R04C02	0.098	0.058	NaN	
101032570152_R04C01	0.058	0.036	NaN	
...	...	...	...	
9996247054_R03C01	0.037	0.040	NaN	
9996247054_R03C02	0.048	0.029	0.155	
9996247055_R03C01	0.042	0.026	NaN	
9996247055_R03C02	0.045	0.025	0.157	
9996247056_R05C02	0.030	0.025	0.052	
IlmnID	ch.X.967194F	ch.X.97129969R	ch.X.97133160R	\
100946230055_R04C01	0.150	0.108	0.076	
100946230056_R04C01	0.149	0.094	0.059	
100946230056_R04C02	0.232	NaN	0.090	
101032570143_R04C02	0.392	NaN	0.206	
101032570152_R04C01	0.350	NaN	0.137	
...	...	...	...	
9996247054_R03C01	0.355	NaN	0.052	
9996247054_R03C02	0.283	NaN	0.065	
9996247055_R03C01	0.395	NaN	0.057	
9996247055_R03C02	0.384	0.143	0.059	
9996247056_R05C02	0.123	0.046	0.045	
IlmnID	ch.X.97651759F	ch.X.97737721F	ch.X.98007042R	
100946230055_R04C01	0.022	0.063	0.078	
100946230056_R04C01	0.019	0.058	0.057	
100946230056_R04C02	0.022	0.073	0.128	
101032570143_R04C02	0.032	0.070	0.142	
101032570152_R04C01	0.031	0.061	0.152	
...	...	...	...	
9996247054_R03C01	0.029	0.055	0.099	
9996247054_R03C02	0.037	0.086	0.180	
9996247055_R03C01	0.031	0.067	0.132	
9996247055_R03C02	0.024	0.055	0.097	
9996247056_R05C02	0.026	0.065	0.063	
[120 rows x 485512 columns]				

## 4.4.5 Drop out Probes with a Percentage of NaNs

If you want to drop the probes with either all NaNs or a percentage of NaNs, use this code below. However, there are some scenarios where you will have to add back those probe columns, so only use this step if you have to.

```
[46]: threshold = 0.95
final_betas = qc_betas.dropna(axis=1, thresh = int(threshold*qc_betas.shape[0]))
print(f'{qc_betas.shape[1] - final_betas.shape[1]} probe(s) removed because of NaNs')
#print(f'Sample(s) removed: {set(qc_betas.columns) - set(final_betas.columns)}')
↪#could be a long output
```

(continues on next page)

(continued from previous page)

final_betas					
40600 probe(s) removed because of NaNs					
[46]:	IlmnID	cg00000029	cg00000108	cg00000236	cg00000292 \
	100946230055_R04C01	0.864	0.971	0.935	0.945
	100946230056_R04C01	0.854	0.978	0.932	0.971
	100946230056_R04C02	0.879	0.958	0.899	0.971
	101032570143_R04C02	0.837	0.968	0.918	0.951
	101032570152_R04C01	0.813	0.971	0.934	0.955
	...	...	...	...	...
	9996247054_R03C01	0.840	0.957	0.917	0.932
	9996247054_R03C02	0.864	0.963	0.885	0.929
	9996247055_R03C01	0.817	0.956	0.887	0.903
	9996247055_R03C02	0.801	0.965	0.893	0.962
	9996247056_R05C02	0.837	0.976	0.957	0.927
	IlmnID	cg00000321	cg00000363	cg00000622	cg00000658 \
	100946230055_R04C01	0.378	0.468	0.010	0.862
	100946230056_R04C01	0.421	0.397	0.012	0.907
	100946230056_R04C02	0.191	0.560	0.012	0.871
	101032570143_R04C02	0.435	0.456	0.011	0.912
	101032570152_R04C01	0.358	0.372	0.011	0.835
	...	...	...	...	...
	9996247054_R03C01	0.374	0.374	0.012	0.904
	9996247054_R03C02	0.393	0.418	0.019	0.884
	9996247055_R03C01	0.423	0.490	0.014	0.871
	9996247055_R03C02	0.340	0.467	0.013	0.892
	9996247056_R05C02	0.402	0.454	0.013	0.912
	IlmnID	cg00000714	cg00000721	...	ch.X.92543860F \
	100946230055_R04C01	0.249	0.936	...	0.026
	100946230056_R04C01	0.278	0.954	...	0.022
	100946230056_R04C02	0.235	0.923	...	0.029
	101032570143_R04C02	0.343	0.938	...	0.033
	101032570152_R04C01	0.333	0.957	...	0.027
	...	...	...	...	...
	9996247054_R03C01	0.411	0.923	...	0.030
	9996247054_R03C02	0.335	0.927	...	0.033
	9996247055_R03C01	0.391	0.894	...	0.026
	9996247055_R03C02	0.309	0.932	...	0.033
	9996247056_R05C02	0.299	0.945	...	0.029
	IlmnID	ch.X.92554290F	ch.X.93511680F	ch.X.938089F	\
	100946230055_R04C01	0.023	0.046	0.036	
	100946230056_R04C01	0.023	0.034	0.040	
	100946230056_R04C02	0.022	0.038	0.057	
	101032570143_R04C02	0.038	0.034	0.098	
	101032570152_R04C01	0.032	0.044	0.058	
	...	...	...	...	
	9996247054_R03C01	0.028	0.050	0.037	
	9996247054_R03C02	0.037	0.047	0.048	
	9996247055_R03C01	0.032	0.037	0.042	
	9996247055_R03C02	0.028	0.046	0.045	
	9996247056_R05C02	0.030	0.049	0.030	
	IlmnID	ch.X.94051109R	ch.X.967194F	ch.X.97133160R	\
	100946230055_R04C01	0.031	0.150	0.076	

(continues on next page)

(continued from previous page)

```

100946230056_R04C01      0.034      0.149      0.059
100946230056_R04C02      0.045      0.232      0.090
101032570143_R04C02      0.058      0.392      0.206
101032570152_R04C01      0.036      0.350      0.137
...
9996247054_R03C01      0.040      0.355      0.052
9996247054_R03C02      0.029      0.283      0.065
9996247055_R03C01      0.026      0.395      0.057
9996247055_R03C02      0.025      0.384      0.059
9996247056_R05C02      0.025      0.123      0.045

IllumID      ch.X.97651759F      ch.X.97737721F      ch.X.98007042R
100946230055_R04C01      0.022      0.063      0.078
100946230056_R04C01      0.019      0.058      0.057
100946230056_R04C02      0.022      0.073      0.128
101032570143_R04C02      0.032      0.070      0.142
101032570152_R04C01      0.031      0.061      0.152
...
9996247054_R03C01      0.029      0.055      0.099
9996247054_R03C02      0.037      0.086      0.180
9996247055_R03C01      0.031      0.067      0.132
9996247055_R03C02      0.024      0.055      0.097
9996247056_R05C02      0.026      0.065      0.063

[120 rows x 444912 columns]
```

Another way to tell if your sample is bad is to [predict the sex](#) of your samples and compare the predicted sex to the actual sex, if that information is available. If the predicted sex does not match the actual sex, this is an indicator that the sample needs to be investigated further, and could potentially be removed.

If you are planning on using your beta values for a machine learning model, you may want to [filter out the sex probes](#) to get rid of any sex bias in your model.

## 4.5 Outlier detection using Multidimensional Scaling (MDS)

Multidimensional scaling (MDS) is one method for comparing the similarity of samples—similar to PCA. MDS based on euclidean distance will also have identical results to PCA, because minimizing linear distance between points is effectively the same as maximizing linear correlation (the latter being what PCA does).

See StatQuest’s very helpful [video](#) on MDS and PCoA for more details.

methyCheck allows users the option of filtering their data based on the results of MDS. The standard cut off is 1.5 standard deviations from the mean of the data, however users have the ability to adjust the cut off value if they are not satisfied by the filtering. After examining the MDS plot, press ‘enter’ to accept the cut-off as it is, or enter a new value and rerun the plot.

We will walk through an example of how to use MDS with this dataset from GEO: [GSE111629](#). This is a large dataset (n=571) of patients with Parkinson’s Disease (n=335) and a group of controls (n=237). These blood samples were run on Illumina’s 450k arrays.

We downloaded the data from GEO and processed it with methyPrep using the following command:

```
>>> python -m methyPrep process -d <filepath> --all
```

**WARNING:** This is a huge dataset and methyPrep will take +8 hours to process it. It will also eat up a lot of storage on your machine. We chose this data because it demonstrates the utility of the MDS function. We recommend

users pick a different dataset to follow along this example!

```
[1]: import methylcheck
import pandas as pd
from pathlib import Path
filepath = Path('/Users/patriciagirardi/methylcheck_tutorial')
```

```
[2]: df = pd.read_pickle('~/.methylcheck_tutorial/beta_values.pkl')
metadata = pd.read_pickle('~/.methylcheck_tutorial/GSE111629_GPL13534_meta_data.pkl')
metadata.head()
```

```
[2]:
```

	GSM_ID	Sample_Name	source	\
0	GSM3035401	genomic DNA from 3999979001_R01C01	X3999979001_R01C01	
1	GSM3035402	genomic DNA from 3999979001_R01C02	X3999979001_R01C02	
2	GSM3035403	genomic DNA from 3999979001_R02C01	X3999979001_R02C01	
3	GSM3035404	genomic DNA from 3999979001_R02C02	X3999979001_R02C02	
4	GSM3035405	genomic DNA from 3999979001_R03C01	X3999979001_R03C01	

	platform	title	disease	state	age	\
0	GPL13534	genomic DNA from 3999979001_R01C01	Parkinson's disease (PD)		74	
1	GPL13534	genomic DNA from 3999979001_R01C02	PD-free control		73	
2	GPL13534	genomic DNA from 3999979001_R02C01	Parkinson's disease (PD)		62	
3	GPL13534	genomic DNA from 3999979001_R02C02	PD-free control		72	
4	GPL13534	genomic DNA from 3999979001_R03C01	PD-free control		72	

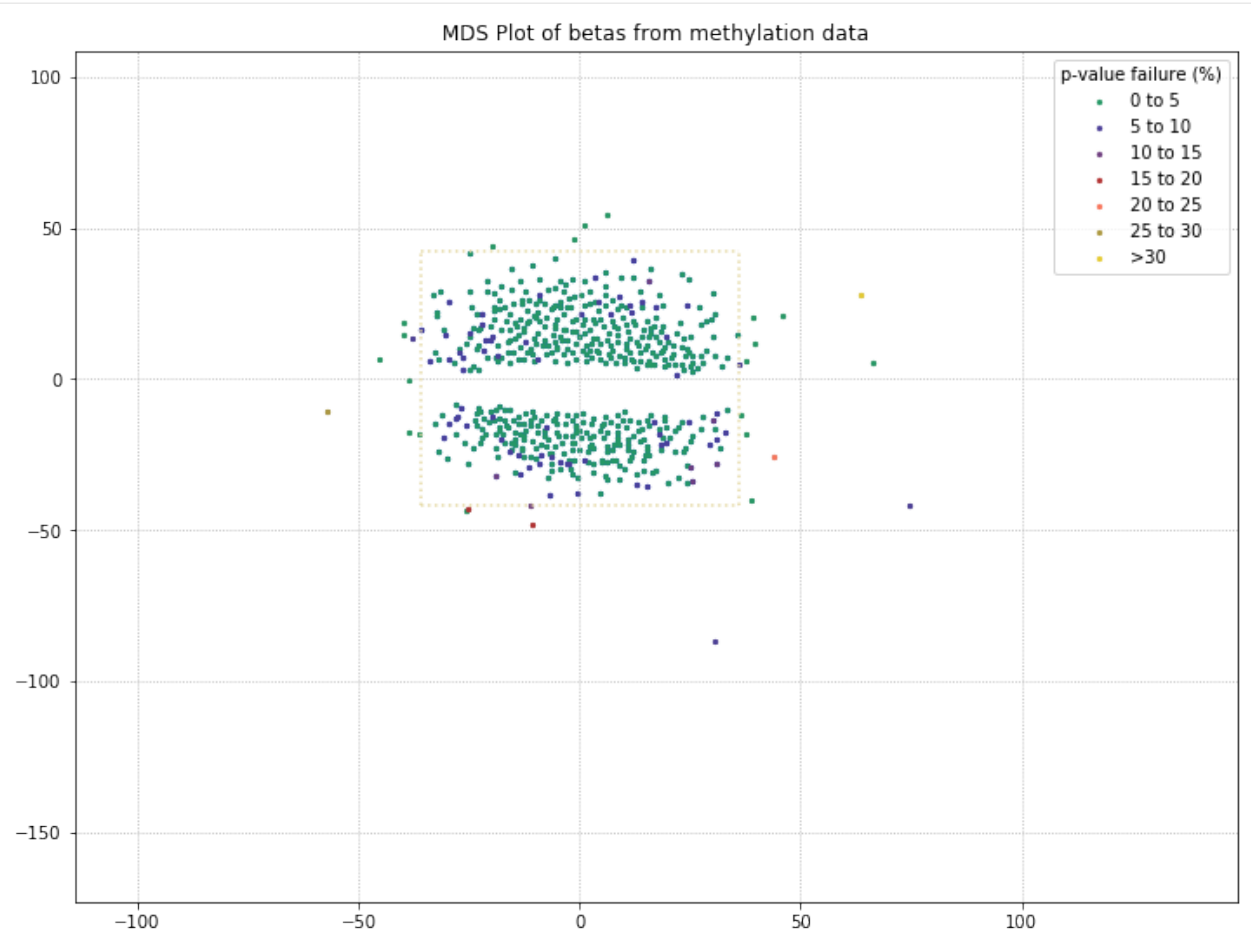
	gender	ethnicity	tissue	Sample_ID	Sentrix_ID	\
0	Female	Caucasian	whole blood	3999979001_R01C01	3999979001	
1	Female	Caucasian	whole blood	3999979001_R01C02	3999979001	
2	Male	Caucasian	whole blood	3999979001_R02C01	3999979001	
3	Male	Caucasian	whole blood	3999979001_R02C02	3999979001	
4	Female	Caucasian	whole blood	3999979001_R03C01	3999979001	

	Sentrix_Position	description
0	R01C01	
1	R01C02	
2	R02C01	
3	R02C02	
4	R03C01	

```
[3]: methylcheck.beta_mds_plot(df, filter_stdev=2, poobah=filepath)
```

```
INFO:methylcheck.samples.postprocessQC:81518.8 probe(s) [avg per sample] were missing,
→ values and removed from MDS calculations; 485512 remaining.
INFO:numexpr.utils:NumExpr defaulting to 8 threads.
```



571 original samples; 542 after filtering  
 Your scale factor was: 2

[3]:

	3999979001_R01C01	3999979001_R01C02	3999979001_R02C01	\
IlmnID				
cg000000029	0.586	0.553	0.586	
cg000000108	0.978	0.974	0.972	
cg000000109	0.917	0.925	0.905	
cg000000165	0.221	0.208	NaN	
cg000000236	0.812	0.881	0.867	
...	...	...	...	
ch.X.97129969R	NaN	0.088	NaN	
ch.X.97133160R	NaN	NaN	NaN	
ch.X.97651759F	0.032	0.030	0.044	
ch.X.97737721F	NaN	NaN	NaN	
ch.X.98007042R	NaN	NaN	NaN	

	3999979001_R02C02	3999979001_R03C01	3999979001_R03C02	\
IlmnID				
cg000000029	0.551	0.533	0.698	
cg000000108	0.973	0.968	0.973	
cg000000109	0.934	0.922	0.920	
cg000000165	0.185	NaN	NaN	
cg000000236	0.888	0.843	0.852	
...	...	...	...	

(continues on next page)

(continued from previous page)

ch.X.97129969R	NaN	0.112	NaN
ch.X.97133160R	NaN	NaN	NaN
ch.X.97651759F	0.031	0.029	0.043
ch.X.97737721F	NaN	NaN	NaN
ch.X.98007042R	NaN	NaN	NaN
3999979001_R04C01 3999979001_R04C02 3999979001_R05C01 \			
IlmnID			
cg00000029	0.713	0.413	0.367
cg00000108	0.971	0.967	0.970
cg00000109	0.913	0.918	0.933
cg00000165	NaN	NaN	0.160
cg00000236	0.889	0.815	0.809
...	...	...	...
ch.X.97129969R	0.102	NaN	NaN
ch.X.97133160R	NaN	NaN	NaN
ch.X.97651759F	0.024	0.038	0.038
ch.X.97737721F	NaN	NaN	NaN
ch.X.98007042R	NaN	NaN	NaN
3999979001_R05C02 ... 9721367028_R01C02 9721367028_R02C02 \			
IlmnID			
cg00000029	0.655	...	0.719
cg00000108	0.976	...	0.974
cg00000109	0.923	...	0.929
cg00000165	0.209	...	0.209
cg00000236	0.887	...	0.888
...	...	...	...
ch.X.97129969R	0.118	...	0.110
ch.X.97133160R	NaN	...	NaN
ch.X.97651759F	0.024	...	0.025
ch.X.97737721F	NaN	...	NaN
ch.X.98007042R	NaN	...	NaN
9721367028_R03C01 9721367028_R03C02 9721367028_R04C01 \			
IlmnID			
cg00000029	0.521	0.643	0.531
cg00000108	0.970	0.973	0.972
cg00000109	0.930	0.924	0.930
cg00000165	NaN	NaN	NaN
cg00000236	0.902	0.889	0.875
...	...	...	...
ch.X.97129969R	0.103	NaN	NaN
ch.X.97133160R	NaN	NaN	NaN
ch.X.97651759F	0.032	0.039	0.041
ch.X.97737721F	NaN	NaN	NaN
ch.X.98007042R	NaN	NaN	NaN
9721367028_R04C02 9721367028_R05C01 9721367028_R05C02 \			
IlmnID			
cg00000029	0.578	0.518	0.606
cg00000108	0.968	0.969	0.975
cg00000109	0.905	0.893	0.912
cg00000165	NaN	NaN	NaN
cg00000236	0.836	0.884	0.868
...	...	...	...
ch.X.97129969R	NaN	0.104	0.077

(continues on next page)

(continued from previous page)

ch.X.97133160R	NaN	NaN	NaN
ch.X.97651759F	0.042	0.027	0.027
ch.X.97737721F	NaN	NaN	NaN
ch.X.98007042R	NaN	NaN	NaN
	9721367028_R06C01	9721367028_R06C02	
IlmnID			
cg00000029	NaN	0.532	
cg00000108	0.966	0.968	
cg00000109	0.908	0.912	
cg00000165	NaN	NaN	
cg00000236	0.882	0.874	
...	...	...	
ch.X.97129969R	NaN	NaN	
ch.X.97133160R	NaN	NaN	
ch.X.97651759F	0.022	0.043	
ch.X.97737721F	NaN	NaN	
ch.X.98007042R	NaN	NaN	
[485512 rows x 542 columns]			

There are a few things to note with the MDS plot shown above. We've color-coded the samples in the plot by their poobah value failure rates; the samples with the highest failure rates (yellow, orange) tend to separate from the two clusters.

The other thing to note is that there are two clusters. With our color-coding scheme, we can only hope that these are the two groups (Parkinson's Disease patients vs. controls). So we need to adjust our color-coding to examine that possibility!

```
[4]: metadata['disease state'].value_counts()
```

```
[4]: Parkinson's disease (PD)      335
     PD-free control              237
     Name: disease state, dtype: int64
```

```
[5]: disease_state = metadata[['Sample_ID', 'disease state']]
     disease_labels = dict(zip(disease_state['Sample_ID'], disease_state['disease state']))
```

There's a small error in the metadata where one of the rows has an empty sample ID column, which will cause an error when we try to plot the mds based on the labels. See below how we call an empty string key from the labels dictionary and still get a value for the disease state. So we'll make sure to drop that key/value pair out from our dictionary before we continue.

```
[6]: disease_labels['']
```

```
[6]: "Parkinson's disease (PD) "
```

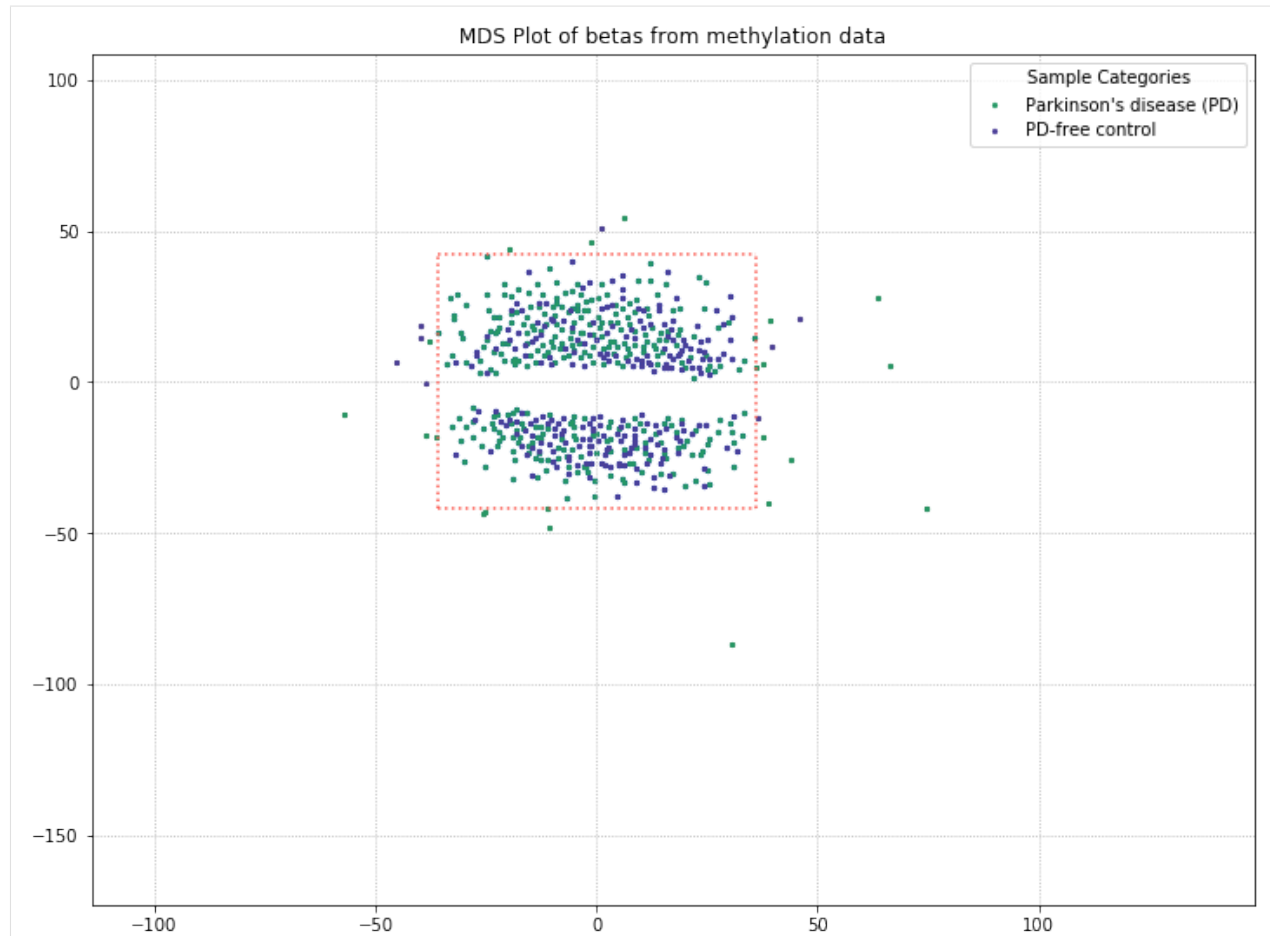
```
[7]: del disease_labels['']
```

Now our dictionary won't error when we try to label samples in our MDS plot, so we can proceed with plotting the control vs. PD color-coding.

```
[8]: methylcheck.beta_mds_plot(df, filter_stdev=2, labels=disease_labels)
```

```
INFO:methylcheck.samples.postprocessQC:81518.8 probe(s) [avg per sample] were missing,
↪ values and removed from MDS calculations; 485512 remaining.
```





571 original samples; 542 after filtering  
Your scale factor was: 2

```
[8]:
```

	3999979001_R01C01	3999979001_R01C02	3999979001_R02C01	\
IlmnID				
cg00000029	0.586	0.553	0.586	
cg00000108	0.978	0.974	0.972	
cg00000109	0.917	0.925	0.905	
cg00000165	0.221	0.208	NaN	
cg00000236	0.812	0.881	0.867	
...	...	...	...	
ch.X.97129969R	NaN	0.088	NaN	
ch.X.97133160R	NaN	NaN	NaN	
ch.X.97651759F	0.032	0.030	0.044	
ch.X.97737721F	NaN	NaN	NaN	
ch.X.98007042R	NaN	NaN	NaN	

	3999979001_R02C02	3999979001_R03C01	3999979001_R03C02	\
IlmnID				
cg00000029	0.551	0.533	0.698	
cg00000108	0.973	0.968	0.973	
cg00000109	0.934	0.922	0.920	
cg00000165	0.185	NaN	NaN	
cg00000236	0.888	0.843	0.852	
...	...	...	...	

(continues on next page)

(continued from previous page)

ch.X.97129969R	NaN	0.112	NaN
ch.X.97133160R	NaN	NaN	NaN
ch.X.97651759F	0.031	0.029	0.043
ch.X.97737721F	NaN	NaN	NaN
ch.X.98007042R	NaN	NaN	NaN
3999979001_R04C01 3999979001_R04C02 3999979001_R05C01 \			
IlmnID			
cg00000029	0.713	0.413	0.367
cg00000108	0.971	0.967	0.970
cg00000109	0.913	0.918	0.933
cg00000165	NaN	NaN	0.160
cg00000236	0.889	0.815	0.809
...	...	...	...
ch.X.97129969R	0.102	NaN	NaN
ch.X.97133160R	NaN	NaN	NaN
ch.X.97651759F	0.024	0.038	0.038
ch.X.97737721F	NaN	NaN	NaN
ch.X.98007042R	NaN	NaN	NaN
3999979001_R05C02 ... 9721367028_R01C02 9721367028_R02C02 \			
IlmnID			
cg00000029	0.655	...	0.719
cg00000108	0.976	...	0.974
cg00000109	0.923	...	0.929
cg00000165	0.209	...	0.209
cg00000236	0.887	...	0.888
...	...	...	...
ch.X.97129969R	0.118	...	0.110
ch.X.97133160R	NaN	...	NaN
ch.X.97651759F	0.024	...	0.025
ch.X.97737721F	NaN	...	NaN
ch.X.98007042R	NaN	...	NaN
9721367028_R03C01 9721367028_R03C02 9721367028_R04C01 \			
IlmnID			
cg00000029	0.521	0.643	0.531
cg00000108	0.970	0.973	0.972
cg00000109	0.930	0.924	0.930
cg00000165	NaN	NaN	NaN
cg00000236	0.902	0.889	0.875
...	...	...	...
ch.X.97129969R	0.103	NaN	NaN
ch.X.97133160R	NaN	NaN	NaN
ch.X.97651759F	0.032	0.039	0.041
ch.X.97737721F	NaN	NaN	NaN
ch.X.98007042R	NaN	NaN	NaN
9721367028_R04C02 9721367028_R05C01 9721367028_R05C02 \			
IlmnID			
cg00000029	0.578	0.518	0.606
cg00000108	0.968	0.969	0.975
cg00000109	0.905	0.893	0.912
cg00000165	NaN	NaN	NaN
cg00000236	0.836	0.884	0.868
...	...	...	...
ch.X.97129969R	NaN	0.104	0.077

(continues on next page)

(continued from previous page)

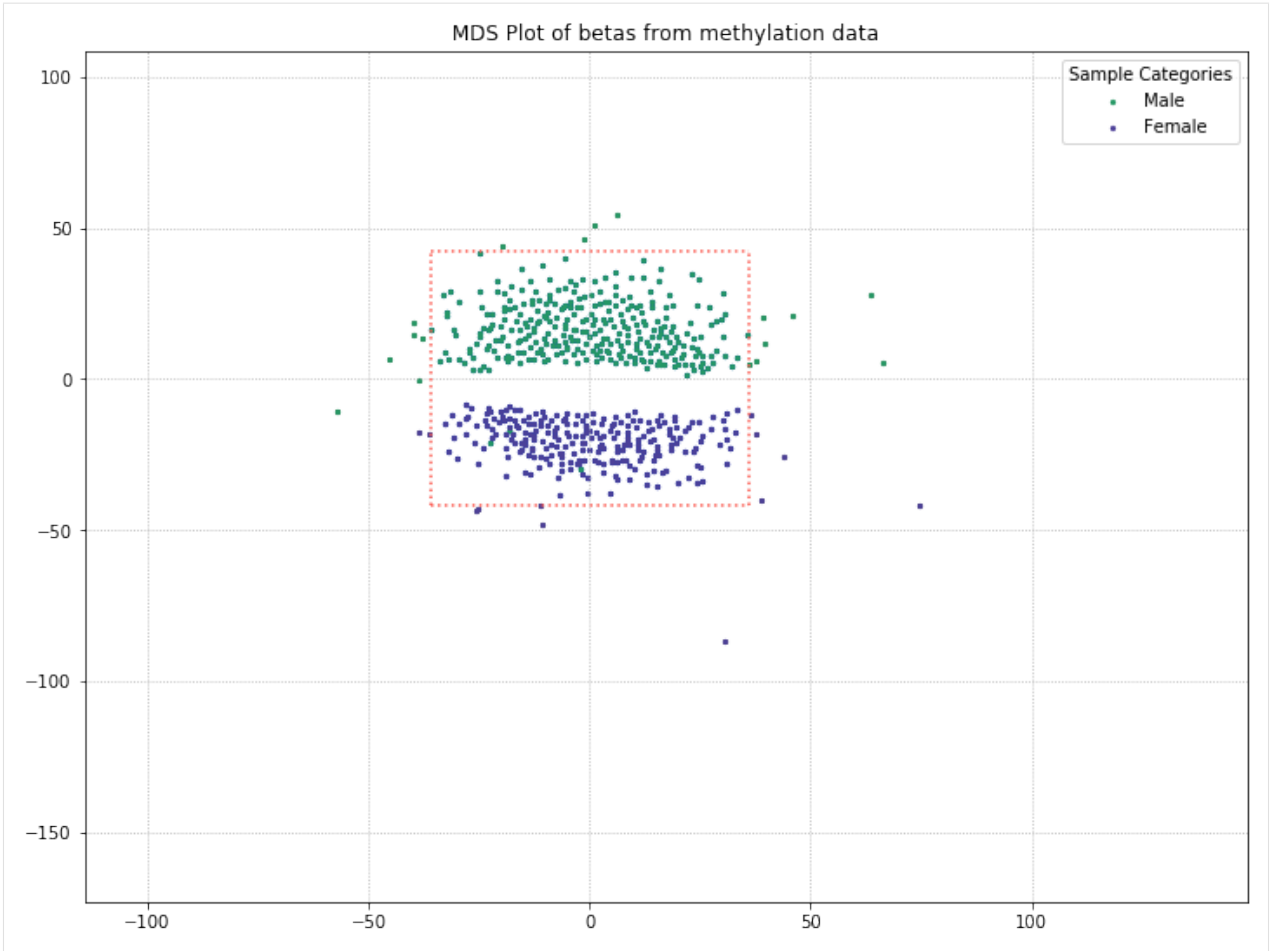
ch.X.97133160R	NaN	NaN	NaN
ch.X.97651759F	0.042	0.027	0.027
ch.X.97737721F	NaN	NaN	NaN
ch.X.98007042R	NaN	NaN	NaN
	9721367028_R06C01	9721367028_R06C02	
IlmnID			
cg00000029	NaN	0.532	
cg00000108	0.966	0.968	
cg00000109	0.908	0.912	
cg00000165	NaN	NaN	
cg00000236	0.882	0.874	
...	...	...	
ch.X.97129969R	NaN	NaN	
ch.X.97133160R	NaN	NaN	
ch.X.97651759F	0.022	0.043	
ch.X.97737721F	NaN	NaN	
ch.X.98007042R	NaN	NaN	
[485512 rows x 542 columns]			

It appears that disease state isn't responsible for the clustering we're seeing here, based on these results. The PD and control patients are evenly interspersed among the two clusters. There might be some other attribute in the metadata responsible for this, though. The best candidate would be something with a binary result in this dataset, like patient gender. So let's examine what the MDS plot looks like when color-coding by patient gender.

```
[9]: gender = metadata[['Sample_ID', 'gender']]
gender_labels = dict(zip(gender['Sample_ID'], gender['gender']))
del gender_labels['']

methylcheck.beta_mds_plot(df, filter_stdev=2, labels=gender_labels)
```

INFO:methylcheck.samples.postprocessQC:81518.8 probe(s) [avg per sample] were missing,
↪ values and removed from MDS calculations; 485512 remaining.



571 original samples; 542 after filtering  
Your scale factor was: 2

[9]:

	3999979001_R01C01	3999979001_R01C02	3999979001_R02C01	\
IlmnID				
cg000000029	0.586	0.553	0.586	
cg000000108	0.978	0.974	0.972	
cg000000109	0.917	0.925	0.905	
cg000000165	0.221	0.208	NaN	
cg000000236	0.812	0.881	0.867	
...	...	...	...	
ch.X.97129969R	NaN	0.088	NaN	
ch.X.97133160R	NaN	NaN	NaN	
ch.X.97651759F	0.032	0.030	0.044	
ch.X.97737721F	NaN	NaN	NaN	
ch.X.98007042R	NaN	NaN	NaN	

	3999979001_R02C02	3999979001_R03C01	3999979001_R03C02	\
IlmnID				
cg000000029	0.551	0.533	0.698	
cg000000108	0.973	0.968	0.973	
cg000000109	0.934	0.922	0.920	
cg000000165	0.185	NaN	NaN	
cg000000236	0.888	0.843	0.852	
...	...	...	...	

(continues on next page)

(continued from previous page)

ch.X.97129969R	NaN	0.112	NaN
ch.X.97133160R	NaN	NaN	NaN
ch.X.97651759F	0.031	0.029	0.043
ch.X.97737721F	NaN	NaN	NaN
ch.X.98007042R	NaN	NaN	NaN
3999979001_R04C01 3999979001_R04C02 3999979001_R05C01 \			
IlmnID			
cg00000029	0.713	0.413	0.367
cg00000108	0.971	0.967	0.970
cg00000109	0.913	0.918	0.933
cg00000165	NaN	NaN	0.160
cg00000236	0.889	0.815	0.809
...	...	...	...
ch.X.97129969R	0.102	NaN	NaN
ch.X.97133160R	NaN	NaN	NaN
ch.X.97651759F	0.024	0.038	0.038
ch.X.97737721F	NaN	NaN	NaN
ch.X.98007042R	NaN	NaN	NaN
3999979001_R05C02 ... 9721367028_R01C02 9721367028_R02C02 \			
IlmnID			
cg00000029	0.655	...	0.719
cg00000108	0.976	...	0.974
cg00000109	0.923	...	0.929
cg00000165	0.209	...	0.209
cg00000236	0.887	...	0.888
...	...	...	...
ch.X.97129969R	0.118	...	0.110
ch.X.97133160R	NaN	...	NaN
ch.X.97651759F	0.024	...	0.025
ch.X.97737721F	NaN	...	NaN
ch.X.98007042R	NaN	...	NaN
9721367028_R03C01 9721367028_R03C02 9721367028_R04C01 \			
IlmnID			
cg00000029	0.521	0.643	0.531
cg00000108	0.970	0.973	0.972
cg00000109	0.930	0.924	0.930
cg00000165	NaN	NaN	NaN
cg00000236	0.902	0.889	0.875
...	...	...	...
ch.X.97129969R	0.103	NaN	NaN
ch.X.97133160R	NaN	NaN	NaN
ch.X.97651759F	0.032	0.039	0.041
ch.X.97737721F	NaN	NaN	NaN
ch.X.98007042R	NaN	NaN	NaN
9721367028_R04C02 9721367028_R05C01 9721367028_R05C02 \			
IlmnID			
cg00000029	0.578	0.518	0.606
cg00000108	0.968	0.969	0.975
cg00000109	0.905	0.893	0.912
cg00000165	NaN	NaN	NaN
cg00000236	0.836	0.884	0.868
...	...	...	...
ch.X.97129969R	NaN	0.104	0.077

(continues on next page)

(continued from previous page)

ch.X.97133160R	NaN	NaN	NaN
ch.X.97651759F	0.042	0.027	0.027
ch.X.97737721F	NaN	NaN	NaN
ch.X.98007042R	NaN	NaN	NaN
	9721367028_R06C01	9721367028_R06C02	
IlmnID			
cg00000029	NaN	0.532	
cg00000108	0.966	0.968	
cg00000109	0.908	0.912	
cg00000165	NaN	NaN	
cg00000236	0.882	0.874	
...	...	...	
ch.X.97129969R	NaN	NaN	
ch.X.97133160R	NaN	NaN	
ch.X.97651759F	0.022	0.043	
ch.X.97737721F	NaN	NaN	
ch.X.98007042R	NaN	NaN	
[485512 rows x 542 columns]			

It looks like this might be the culprit of the two clusters in the MDS plot! There are a few outliers, but for the most part, the two clusters are made up of one gender each.

Utilizing one of the functions covered in our Filtering Probes section, we can pull out the beta values from X/Y probes and just examine beta values from probes on the autosomal chromosomes. If MDS is picking up on differences between patient sex and clustering the patients accordingly, we might be able to get at an underlying difference by removing sex from the equation.

```
[11]: no_sex_probes = methylcheck.exclude_sex_control_probes(df, array='450k', no_sex=True,
↳no_control=True)

methylcheck.beta_mds_plot(no_sex_probes, filter_stdev=2, labels=gender_labels)

INFO:methylcheck.samples.postprocessQC:79013.0 probe(s) [avg per sample] were missing
↳values and removed from MDS calculations; 473864 remaining.
```

	3999979001_R01C01	3999979001_R01C02	3999979001_R02C01	\
IlmnID				
cg00000029	0.586	0.553	0.586	
cg00000108	0.978	0.974	0.972	
cg00000109	0.917	0.925	0.905	
cg00000165	0.221	0.208	NaN	
cg00000236	0.812	0.881	0.867	
...	...	...	...	
ch.9.98937537R	NaN	NaN	NaN	
ch.9.98957343R	NaN	NaN	NaN	
ch.9.98959675F	NaN	NaN	NaN	
ch.9.98989607R	NaN	NaN	NaN	
ch.9.991104F	NaN	NaN	NaN	
	3999979001_R02C02	3999979001_R03C01	3999979001_R03C02	\
IlmnID				
cg00000029	0.551	0.533	0.698	
cg00000108	0.973	0.968	0.973	
cg00000109	0.934	0.922	0.920	
cg00000165	0.185	NaN	NaN	
cg00000236	0.888	0.843	0.852	
...	...	...	...	

#### 4.5. Outlier detection using Multidimensional Scaling (MDS)

(continued from previous page)

ch.9.98937537R	NaN	NaN	NaN
ch.9.98957343R	NaN	NaN	NaN
ch.9.98959675F	NaN	NaN	NaN
ch.9.98989607R	NaN	NaN	NaN
ch.9.991104F	NaN	NaN	NaN
3999979001_R04C01 3999979001_R05C01 3999979001_R05C02 \			
IlmnID			
cg00000029	0.713	0.367	0.655
cg00000108	0.971	0.970	0.976
cg00000109	0.913	0.933	0.923
cg00000165	NaN	0.160	0.209
cg00000236	0.889	0.809	0.887
...	...	...	...
ch.9.98937537R	NaN	NaN	NaN
ch.9.98957343R	NaN	NaN	NaN
ch.9.98959675F	NaN	NaN	NaN
ch.9.98989607R	NaN	NaN	NaN
ch.9.991104F	NaN	NaN	NaN
3999979001_R06C01 ... 9721367028_R01C02 9721367028_R02C02 \			
IlmnID			
cg00000029	0.695	...	0.719
cg00000108	0.976	...	0.974
cg00000109	0.925	...	0.929
cg00000165	0.265	...	0.209
cg00000236	0.880	...	0.888
...	...	...	...
ch.9.98937537R	NaN	...	NaN
ch.9.98957343R	NaN	...	NaN
ch.9.98959675F	NaN	...	NaN
ch.9.98989607R	NaN	...	NaN
ch.9.991104F	NaN	...	NaN
9721367028_R03C01 9721367028_R03C02 9721367028_R04C01 \			
IlmnID			
cg00000029	0.521	0.643	0.531
cg00000108	0.970	0.973	0.972
cg00000109	0.930	0.924	0.930
cg00000165	NaN	NaN	NaN
cg00000236	0.902	0.889	0.875
...	...	...	...
ch.9.98937537R	NaN	NaN	NaN
ch.9.98957343R	NaN	NaN	NaN
ch.9.98959675F	NaN	NaN	NaN
ch.9.98989607R	NaN	NaN	NaN
ch.9.991104F	NaN	NaN	NaN
9721367028_R04C02 9721367028_R05C01 9721367028_R05C02 \			
IlmnID			
cg00000029	0.578	0.518	0.606
cg00000108	0.968	0.969	0.975
cg00000109	0.905	0.893	0.912
cg00000165	NaN	NaN	NaN
cg00000236	0.836	0.884	0.868
...	...	...	...
ch.9.98937537R	NaN	NaN	NaN

(continues on next page)



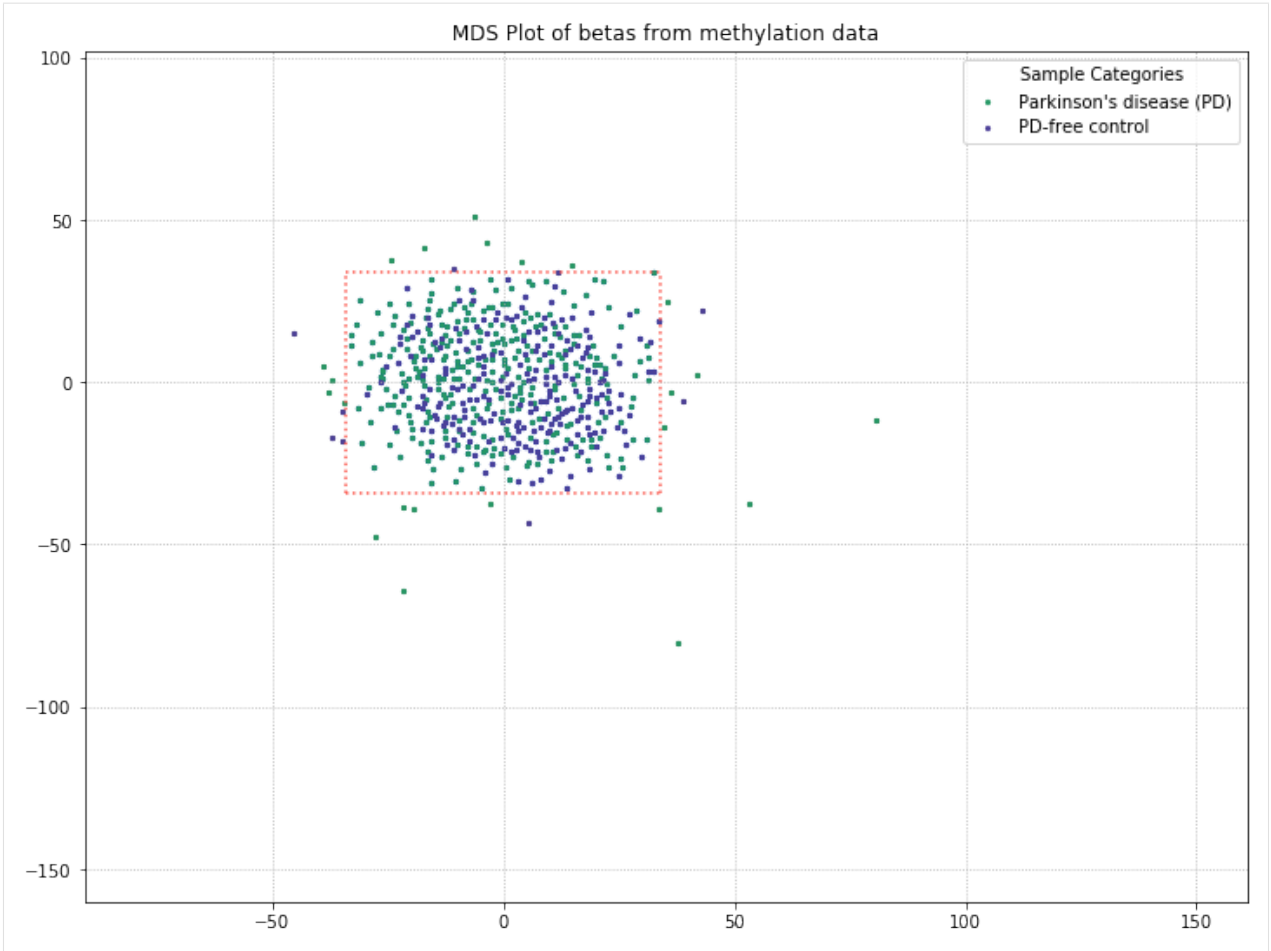
(continued from previous page)

ch.9.98957343R	NaN	NaN	NaN
ch.9.98959675F	NaN	NaN	NaN
ch.9.98989607R	NaN	NaN	NaN
ch.9.991104F	NaN	NaN	NaN
	9721367028_R06C01	9721367028_R06C02	
IlmnID			
cg00000029	NaN	0.532	
cg00000108	0.966	0.968	
cg00000109	0.908	0.912	
cg00000165	NaN	NaN	
cg00000236	0.882	0.874	
...	...	...	
ch.9.98937537R	NaN	NaN	
ch.9.98957343R	NaN	NaN	
ch.9.98959675F	NaN	NaN	
ch.9.98989607R	NaN	NaN	
ch.9.991104F	NaN	NaN	
[473864 rows x 539 columns]			

We lost the separation between clusters as a result of pulling out sex probes. There's still a chance that there is some kind of grouping going on with PD/control patients (for example, the top half of the data is all PD patients, while the bottom half is all control patients). So we'll color-code by disease state to verify whether or not there's any grouping going on.

```
[12]: methylcheck.beta_mds_plot(no_sex_probes, filter_stdev=2, labels=disease_labels)
```

```
INFO:methylcheck.samples.postprocessQC:79013.0 probe(s) [avg per sample] were missing,
↪ values and removed from MDS calculations; 473864 remaining.
```



[12]:

	3999979001_R01C01	3999979001_R01C02	3999979001_R02C01	\
IlmnID				
cg000000029	0.586	0.553	0.586	
cg000000108	0.978	0.974	0.972	
cg000000109	0.917	0.925	0.905	
cg000000165	0.221	0.208	NaN	
cg000000236	0.812	0.881	0.867	
...	...	...	...	
ch.9.98937537R	NaN	NaN	NaN	
ch.9.98957343R	NaN	NaN	NaN	
ch.9.98959675F	NaN	NaN	NaN	
ch.9.98989607R	NaN	NaN	NaN	
ch.9.991104F	NaN	NaN	NaN	
	3999979001_R02C02	3999979001_R03C01	3999979001_R03C02	\
IlmnID				
cg000000029	0.551	0.533	0.698	
cg000000108	0.973	0.968	0.973	
cg000000109	0.934	0.922	0.920	
cg000000165	0.185	NaN	NaN	
cg000000236	0.888	0.843	0.852	
...	...	...	...	

(continues on next page)

(continued from previous page)

ch.9.98937537R	NaN	NaN	NaN
ch.9.98957343R	NaN	NaN	NaN
ch.9.98959675F	NaN	NaN	NaN
ch.9.98989607R	NaN	NaN	NaN
ch.9.991104F	NaN	NaN	NaN
3999979001_R04C01 3999979001_R05C01 3999979001_R05C02 \			
IlmnID			
cg00000029	0.713	0.367	0.655
cg00000108	0.971	0.970	0.976
cg00000109	0.913	0.933	0.923
cg00000165	NaN	0.160	0.209
cg00000236	0.889	0.809	0.887
...	...	...	...
ch.9.98937537R	NaN	NaN	NaN
ch.9.98957343R	NaN	NaN	NaN
ch.9.98959675F	NaN	NaN	NaN
ch.9.98989607R	NaN	NaN	NaN
ch.9.991104F	NaN	NaN	NaN
3999979001_R06C01 ... 9721367028_R01C02 9721367028_R02C02 \			
IlmnID			
cg00000029	0.695	...	0.719
cg00000108	0.976	...	0.974
cg00000109	0.925	...	0.929
cg00000165	0.265	...	0.209
cg00000236	0.880	...	0.888
...	...	...	...
ch.9.98937537R	NaN	...	NaN
ch.9.98957343R	NaN	...	NaN
ch.9.98959675F	NaN	...	NaN
ch.9.98989607R	NaN	...	NaN
ch.9.991104F	NaN	...	NaN
9721367028_R03C01 9721367028_R03C02 9721367028_R04C01 \			
IlmnID			
cg00000029	0.521	0.643	0.531
cg00000108	0.970	0.973	0.972
cg00000109	0.930	0.924	0.930
cg00000165	NaN	NaN	NaN
cg00000236	0.902	0.889	0.875
...	...	...	...
ch.9.98937537R	NaN	NaN	NaN
ch.9.98957343R	NaN	NaN	NaN
ch.9.98959675F	NaN	NaN	NaN
ch.9.98989607R	NaN	NaN	NaN
ch.9.991104F	NaN	NaN	NaN
9721367028_R04C02 9721367028_R05C01 9721367028_R05C02 \			
IlmnID			
cg00000029	0.578	0.518	0.606
cg00000108	0.968	0.969	0.975
cg00000109	0.905	0.893	0.912
cg00000165	NaN	NaN	NaN
cg00000236	0.836	0.884	0.868
...	...	...	...
ch.9.98937537R	NaN	NaN	NaN

(continues on next page)

(continued from previous page)

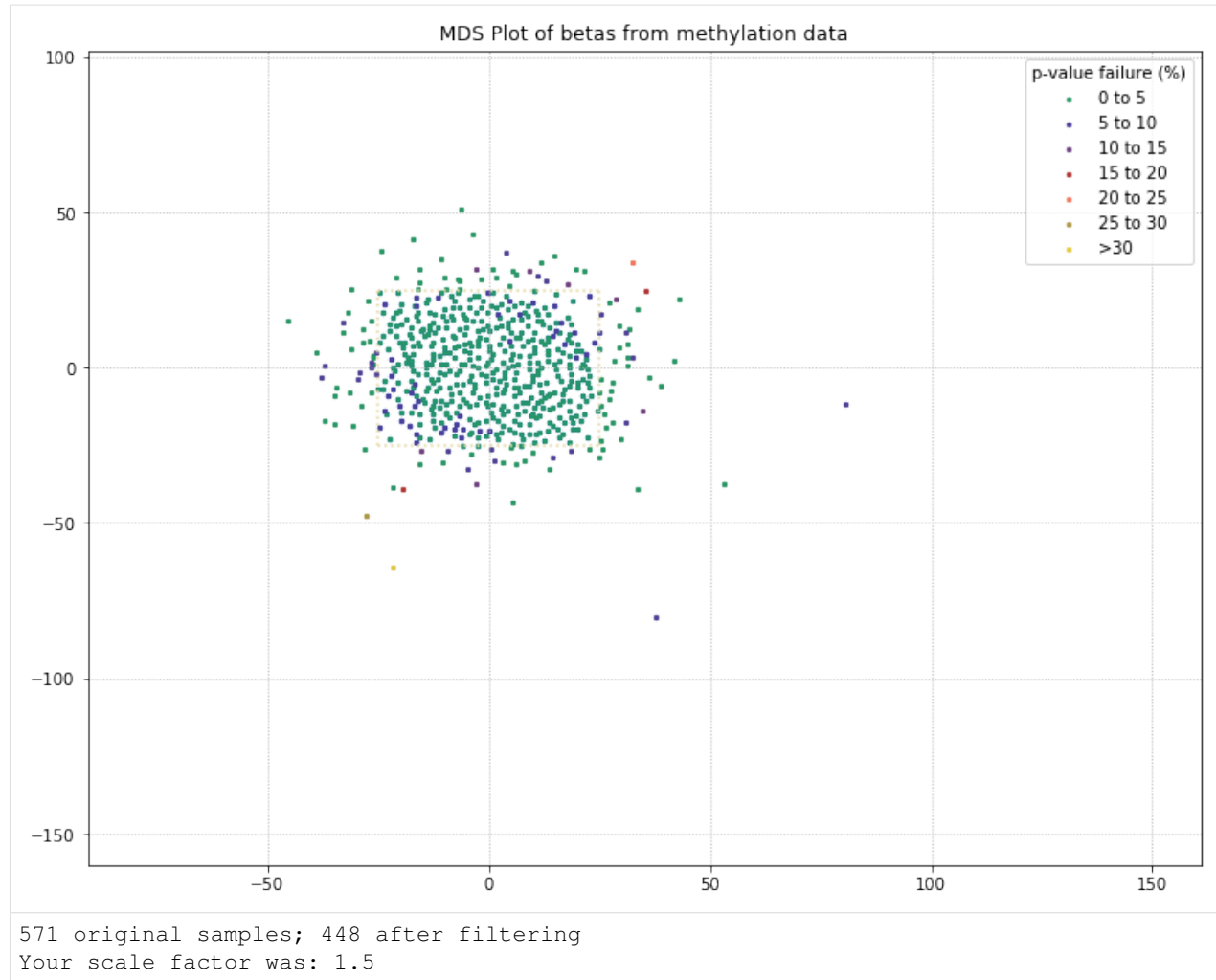
ch.9.98957343R	NaN	NaN	NaN
ch.9.98959675F	NaN	NaN	NaN
ch.9.98989607R	NaN	NaN	NaN
ch.9.991104F	NaN	NaN	NaN
	9721367028_R06C01	9721367028_R06C02	
IlmnID			
cg00000029	NaN	0.532	
cg00000108	0.966	0.968	
cg00000109	0.908	0.912	
cg00000165	NaN	NaN	
cg00000236	0.882	0.874	
...	...	...	
ch.9.98937537R	NaN	NaN	
ch.9.98957343R	NaN	NaN	
ch.9.98959675F	NaN	NaN	
ch.9.98989607R	NaN	NaN	
ch.9.991104F	NaN	NaN	
[473864 rows x 539 columns]			

It appears that the differences between these two disease states aren't being picked up by MDS in a meaningful way. We'll do one more MDS plot on these filtered probes to check how many of those outlier samples have high detection p-value failure rates.

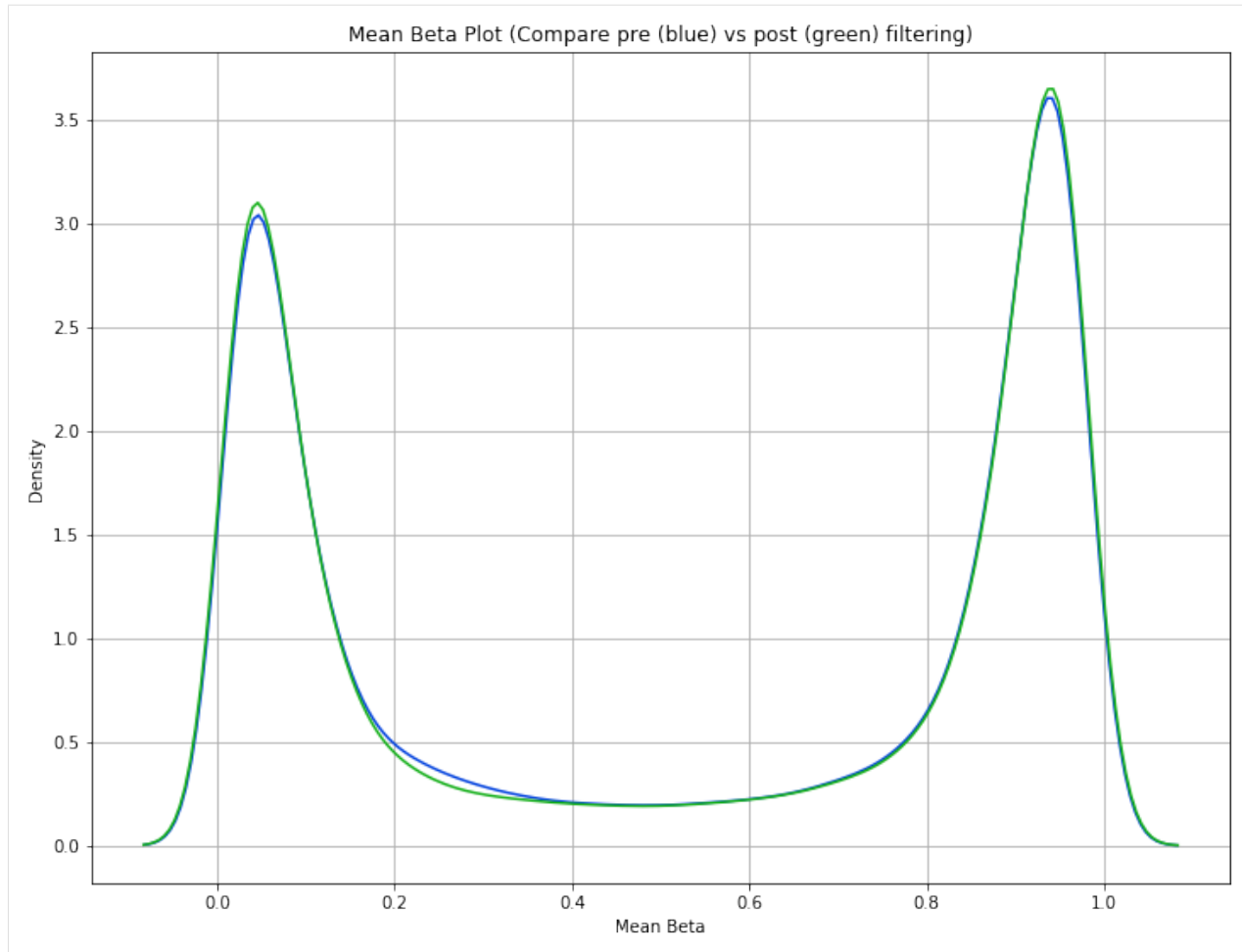
Also, we can define a variable here to access the samples that pass after they've been filtered by MDS. Then, we can compare that beta distribution to the original beta distribution to see how the MDS filtering affected it.

```
[13]: mds_filtered = methyIcheck.beta_mds_plot(no_sex_probes, filter_stdev=1.5,
↳ poobah=filepath)

INFO:methyIcheck.samples.postprocessQC:79013.0 probe(s) [avg per sample] were missing
↳ values and removed from MDS calculations; 473864 remaining.
```



```
[14]: methylcheck.mean_beta_compare(df, mds_filtered)
```



The difference is a bit subtle since even the low quality data in this dataset is relatively high quality, but you can see that the green curve (MDS filtered samples) has higher and tighter peaks than the blue curve (unfiltered data). This means we were able to eliminate some of the samples that had a lot of beta values falling in that midrange between the peaks.

## 4.6 API Reference

<code>methylcheck.cli</code>	
<code>methylcheck.run_pipeline(df, **kwargs)</code>	Run a variety of probe and sample filters in tandem, then plot results
<code>methylcheck.run_qc(path)</code>	Generates all QC plots for a dataset in the path provided.
<code>methylcheck.read_geo(filepath[, verbose, ...])</code>	Use to load preprocessed GEO data into methylcheck. Attempts to find the sample beta/M_values
<code>methylcheck.load(filepath, format, ...)</code>	Methylsuite's all-purpose data loading function.
<code>methylcheck.load_both(filepath, format, ...)</code>	Creates and returns TWO objects (data and meta_data) from the given filepath.
<code>methylcheck.qc_signal_intensity([...])</code>	Suggests sample outliers based on methylated and unmethylated signal intensity.
<code>methylcheck.plot_M_vs_U([...])</code>	plot methylated vs unmethylated probe intensities

Continued on next page

Table 1 – continued from previous page

<code>methylcheck.plot_controls([path, subset, ...])</code>	internal array QC controls (available with the <code>-save_control</code> or <code>-all</code> methylprep process option)
<code>methylcheck.plot_beta_by_type(beta_df[, ...])</code>	compare betas for type I and II probes – (inspired by the <code>plotBetasByType()</code> function)
<i>methylcheck.probes</i>	
<code>methylcheck.list_problem_probes(array[, ...])</code>	Function to create a list of probes to exclude from downstream processes.
<code>methylcheck.exclude_probes(df, probe_list)</code>	Exclude probes from a dataframe of sample beta values.
<code>methylcheck.exclude_sex_control_probes(df, array)</code>	Exclude probes from an array, and return a filtered array.
<code>methylcheck.drop_nan_probes(df[, silent, ...])</code>	accounts for df shape (probes in rows or cols) so <code>dropna()</code> will work.
<i>methylcheck.samples</i>	
<code>methylcheck.sample_plot(df, **kwargs)</code>	A more intuitive alias of <code>beta_density_plot()</code> , since not all values are beta distributions.
<code>methylcheck.beta_density_plot(df[, verbose, ...])</code>	Returns a plot of beta values for each sample in a batch of samples as a separate line.
<code>methylcheck.mean_beta_plot(df[, verbose, ...])</code>	Returns a plot of the average beta values for all probes in a batch of samples.
<code>methylcheck.mean_beta_compare(df1, df2[, ...])</code>	Use this function to compare two dataframes, pre-vs-post filtering and removal of outliers.
<code>methylcheck.beta_mds_plot(df[, ...])</code>	Performs multidimensional scaling on a dataframe of samples
<code>methylcheck.combine_mds(*args, **kwargs)</code>	To combine (or segment) datasets for multidimensional scaling analysis
<code>methylcheck.cumulative_sum_beta_distribution(df)</code>	Attempt to filter outlier samples based on the cumulative area under the curve exceeding a reasonable value (cutoff).
<i>methylcheck.predict</i>	
<code>methylcheck.get_sex(data_source[, ...])</code>	This will calculate and predict the sex of each sample.
<code>methylcheck.assign(df[, two_pass])</code>	Manually and interactively assign each sample to a group, based on beta-value distribution shape.

## 4.6.1 methylcheck.cli

### Functions

<code>build_parser()</code>	
<code>cli_ReportPDF(cmd_args)</code>	
<code>cli_app()</code>	
<code>cli_controls_report(cmd_args)</code>	
<code>cli_qc(cmd_args)</code>	
<code>detect_array(df[, returns, on_lambda])</code>	Determines array type using number of probes columns in df.

### Classes

<code>DefaultParser([prog, usage, description, ...])</code>	
---	--

### 4.6.2 methylcheck.run\_pipeline

`methylcheck.run_pipeline(df, **kwargs)`

Run a variety of probe and sample filters in tandem, then plot results

by specifying all of your options at once, instead of running every part of methylcheck in piecemeal fashion.

this is analogous to using the methylcheck CLI, but for notebooks/scripts

**required:**

**df: (required)**

- data as a DataFrame of beta values (or DataFrame of m\_values)
- sample names in columns and probes in rows

**parameters:**

**verbose: (True/False)** default: False – shows extra info about processing if True

**silent: (True/False)** default: False – suppresses all warnings/info

**exclude\_sex:** filters out probes on sex-chromosomes

**exclude\_control:** filters out illumina control probes

**exclude\_all:** filters out the most probes (sex-linked, control, and all sketchy-listed probes from papers)

**exclude:** (list of strings, shorthand references to papers with sketchy probes to exclude)

**If the array is 450K the publications may include:** 'Chen2013' 'Price2013'  
'Zhou2016' 'Naeem2014' 'DacaRoszak2015'

**If the array is EPIC the publications may include:** 'Zhou2016' 'McCartney2016'

**or these reasons:** 'Polymorphism' 'CrossHybridization'  
'BaseColorChange' 'RepeatSequenceElements'

**or use 'exclude\_all':** to do maximum filtering, including all of these papers' lists.

**plot: (list of strings)** ['mean\_beta\_plot', 'beta\_density\_plot', 'cumulative\_sum\_beta\_distribution', 'beta\_mds\_plot', 'all'] if 'all', then all of these plots will be generated. if omitted, no plots are created.

**save\_plots: (True/False)** default: False

**export (True/False):** default: False – will export the filtered df as a pkl file if True

**note:** this pipeline cannot also apply the array-level `methylcheck.run_qc()` function because that relies on additional probe information that may not be present. Everything in this pipeline applies to a dataframe of beta or m-values for a set of samples.

**returns:** a filtered dataframe object

### 4.6.3 methylcheck.run\_qc

`methylcheck.run_qc(path)`

Generates all QC plots for a dataset in the path provided. if *process -all* was used to create control probes and raw values for QC, because it uses four output files:

- beta\_values.pkl
- control\_probes.pkl



- meth\_values.pkl or noob\_meth\_values.pkl
- unmeth\_values.pkl or noob\_unmeth\_values.pkl

output is all to screen, so best to use in a jupyter notebook. If you prefer output in a PDF, use ReportPDF instead.

Note: this will only look in the path folder; it doesn't do a recursive search for matching files.

#### 4.6.4 methylcheck.read\_geo

`methylcheck.read_geo` (*filepath*, *verbose=False*, *debug=False*, *as\_beta=True*, *column\_pattern=None*, *test\_only=False*, *rename\_probe\_column=True*, *decimals=3*)

**Use to load preprocessed GEO data into methylcheck. Attempts to find the sample beta/M\_values in the CSV/TXT/XLSX file and turn it into a clean dataframe, with probe ids in the index/rows. Version 3 (introduced June 2020)**

- reads a downloaded file, either in csv, xlsx, pickle, txt
- looks for /d\_RxxCxx patterned headings and an probe index
- sets index in df to probes
- sets columns to sample names
- forces probe values to be floats, if strings/mixed
- if filename has 'intensit' or 'signal' in it, this converts to betas and saves even if filename doesn't match, if columns have Methylated in them, it will convert and save
- detect multi-line headers and adjusts dataframe columns accordingly
- returns the usable dataframe

`as_beta == True` – converts meth/unmeth into a df of sample betas. `column_pattern=None` (Sample21 | Sample\_21 | Sample 21) – some string of characters that precedes the number part of each sample in the columns of the file to be ingested.

**FIXED:** [x] handle files with .Signal\_A and .Signal\_B instead of Meth/Unmeth [x] BUG: can't parse **ma-trix**... files if uses underscores instead of spaces around sample numbers, or where sampleXXX has no separator. [x] handle processed files with sample\_XX [x] returns IlmnID as index/probe column, unless 'rename\_probe\_column' == False [x] pass in sample\_column names from header parser so that logic is in one place

(makes the output much larger, so add kwarg to exclude this)

[x] demicals (default 3) – round all probe beta/intensity/p values returned to this number of decimal places.

[x] bug: can only recognize beta samples if 'sample' in column name, or sentrix\_id pattern matches columns.

need to expand this to handle arbitrary sample naming styles (limited to one column per sample patterns)

**TODO:** [-] BUG: meth\_unmeth\_pval works *as\_beta* but not returning full data yet [-] multiline header not working with all files yet. [-] \_family GSM123456-tbl-1.txt files not detected yet

**notes:** this makes inferences based on strings in the filename, and based on the column names.

### 4.6.5 methylcheck.load

`methylcheck.load` (*filepath='.', format='beta\_value', file\_stem='', verbose=False, silent=False, column\_names=None, no\_poobah=False, pval\_cutoff=0.05, no\_filter=True*)

Methylsuite's all-purpose data loading function.

When methylprep processes large datasets, you use the 'batch\_size' option to keep memory and file size more manageable. Use the *load* helper function to quickly load and combine all of those parts into a single data frame of beta-values or m-values.

Doing this with pandas is about 8 times slower than using numpy in the intermediate step.

If no arguments are supplied, it will load all files in current directory that have a 'beta\_values\_X.pkl' pattern.

#### Arguments:

**filepath:** Where to look for all the pickle files of processed data.

**format:** ('beta\_value', 'm\_value', 'meth', 'meth\_df', 'noob\_df', 'beta\_csv', 'sesame') This also allows processed.csv file data to be loaded. If you need meth and unmeth values, choose 'meth' and it will return a data\_containers object with the 'meth' and 'unmeth' values, exactly like the data\_containers object returned by methylprep.run\_pipeline.

If you choose 'meth\_df' or 'noob\_df' it will load the pickled meth and unmeth dataframes from the folder specified.

**column\_names:** if your csv files contain column names that differ from those expected, you can specify them as a list of strings by default it looks for ['noob\_meth', 'noob\_unmeth'] or ['meth', 'unmeth'] or ['beta\_value'] or ['m\_value'] Note: if you csv data has probe names in a column that is not the FIRST column, or is not named "IllumID", you should specify it with column\_names and put it first in the list, like ['illumina\_id', 'noob\_meth', 'noob\_unmeth'].

**no\_poobah:** if loading from CSVs, and there is a column for probe p-values (the poobah\_pval column), the default is to filter out probes that fail the  $p < 0.05$  cutoff. if you specify 'no\_poobah'=True, it will load everything, regardless of p-values.

**pval\_cutoff:** if applying poobah (pvalue probe detection based on poor signal to noise) this specifies the threshold for cutoff (0.05 by default)

**no\_filter: (default = True)** if False, removes probes that illumina, the manufacturer, claimed are sketchy in 2019 for a select list of newer EPIC Sentrix\_IDs. only affects 'beta\_value' and 'm\_value' output; no effect on meth/unmeth raw/NOOB intensity values returned.

**file\_stem: (string)** Older versions (pre v1.3.0) of methylprep processed with batch\_size created a bunch of generically named files, such as 'beta\_values\_1.pkl', 'beta\_values\_2.pkl', 'beta\_values\_3.pkl', and so on. IF you rename these or provide a custom name during processing, provide that name here to load them all. (i.e. if your pickle file is called 'GSE150999\_beta\_values\_X.pkl', then your file\_stem is '**GSE150999\_**')

**verbose:** outputs more processing messages.

**silent:** suppresses all processing messages, even warnings.

#### Use cases and format:

**format = beta\_value:** you have beta\_values.pkl file in the path specified and want a dataframe returned or you have a bunch of beta\_values\_1.pkl files in the path and want them merged and returned as one dataframe (when using 'batch\_size' option in methylprep.run\_pipeline() you'll get multiple files saved)

**format = m\_value:** you have m\_values.pkl file in the path specified and want a dataframe returned or you have a bunch of m\_values\_1.pkl files in the path and want them merged and returned as one dataframe

**format = meth:** (**data\_containers**) you have processed CSV files in the path specified and want a `data_container` returned

**format = meth\_df:** (**dataframe**) you have processed CSV files in the path specified and want a dataframe returned take the `data_containers` object returned and run `methylcheck.container_to_pkl(containers, save=True)` function on it.

**format = noob\_df:** (**dataframe**) loads `noob_meth_values.pkl` and `noob_unmeth_values.pkl` and returns two dataframes in a list

**format = sesame:** for reading csvs processed using R's sesame package. It has a different format (Probe\_ID, ind\_beta, ind\_negs, ind\_poob) per sample. Only those probes that pass the p-value cutoff will be included.

**format = beta\_csv:** for reading processed.csv files from methylprep, and forcing it NOT to load from the pickled beta dataframe file, if present.

**format = poobah\_csv:** similar to `beta_csv`, this pulls poobah p-values for all probes out of all processed CSV files into one dataframe. These p-values will include failed probes and probes that would be filtered by `quality_mask`. 'poobah' excludes these.

**format = poobah:** reads the 'poobah\_values.pkl' file and returns a dataframe of p-values. Note failed / poor-quality probes are replaced with NaN.

---

**Note:**

**Science on p-value cutoff:** This function defaults to a p-value cutoff of 0.05, which is typical for scientific tests. There is currently no consensus on what percent of a sample's probes can fail. For example, if a sample has 860,000 probes and 5% of them fail, should you reject the whole sample from the batch? For large batch industrial scale testing, the authors assign some limit, like 5%, 10%, 20%, 30%, etc as a cutoff. And methylcheck's `run_qc()` function defaults to 10 percent. But the academics we spoke to don't automatically throw out any samples. Because it depends. Cancer samples have lots of aneuploidy (an abnormal number of chromosomes in a haploid set) and lost chromosomes, so one would expect no signal for these CpG sites. So those researchers wouldn't throw out samples unless most of the sample fails. People are working on deriving a calibration curve from public GEO data as a guide, and give a frame of reference, but none exist yet. And public data rarely includes failed samples.

---

---

**Note:**

- modified this from methylprep on 2020-02-20 to allow for `data_containers` to be returned as option
  - v0.6.3: added 'no\_filter' step that automatically removes probes that illumina, the manufacturer, claims are sketchy for certain Catalog IDs. (Disable this with `no_filter=True`)
- 

#### 4.6.6 methylcheck.load\_both

```
methylcheck.load_both(filepath='.', format='beta_value', file_stem="", verbose=False,
                      silent=False, column_names=None, rename_samples=False, sam-
                      ple_names='Sample_Name')
```

Creates and returns TWO objects (data and meta\_data) from the given filepath. Confirms sample names match.

Returns TWO objects (data, meta) as dataframes for analysis. If meta\_data files are found in multiple folders, it will read them all and try to match to the samples in the beta\_values pickles by sample ID.

**Arguments:**

**filepath:** Where to look for all the pickle files of processed data.

**format:** ‘beta\_values’, ‘m\_value’, or some other custom file pattern.

**file\_stem (string):** By default, methylprep process with batch\_size creates a bunch of generically named files, such as ‘beta\_values\_1.pkl’, ‘beta\_values\_2.pkl’, ‘beta\_values\_3.pkl’, and so on. IF you rename these or provide a custom name during processing, provide that name here. (i.e. if your pickle file is called ‘GSE150999\_beta\_values\_X.pkl’, then your file\_stem is ‘**GSE150999\_**’)

**column\_names:** if your processed csv files contain column names that differ from those expected, you can specify them as a list of strings by default it looks for [‘noob\_meth’, ‘noob\_unmeth’] or [‘meth’, ‘unmeth’] or [‘beta\_value’] or [‘m\_value’] Note: if you csv data has probe names in a column that is not the FIRST column, or is not named “IlmnID”, you should specify it with column\_names and put it first in the list, like [‘illumina\_id’, ‘noob\_meth’, ‘noob\_umeth’].

**rename\_samples:** if your meta\_data contains a ‘Sample\_Name’ column, the returned data and meta\_data will have index and columns renamed to Sample\_Names instead of Sample\_IDs, respectively.

**sample\_name (string):** the column name to use in meta dataframe for sample names. Assumes ‘Sample\_Name’ if unspecified.

**verbose:** outputs more processing messages.

**silent:** suppresses all processing messages, even warnings.

#### 4.6.7 methyIcheck.qc\_signal\_intensity

`methyIcheck.qc_signal_intensity` (*data\_containers=None, path=None, meth=None, unmeth=None, poobah=None, palette=None, noob=True, silent=False, verbose=False, plot=True, cutoff\_line=True, bad\_sample\_cutoff=11.5, return\_fig=False*)

Suggests sample outliers based on methylated and unmethylated signal intensity.

**path** to csv files processed using methylprep these have “noob\_meth” and “noob\_unmeth” columns per sample file this function can use. if you want it to processed data uncorrected data.

**data\_containers** output from the methylprep.run\_pipeline() command when run in a script or notebook. you can also recreate the list of datacontainers using `methyIcheck.load(<filepath>,’meth’)`

**(meth and unmeth)** if you chose *process -all* you can load the raw intensities like this, and pass them in: `meth = pd.read_pickle(‘meth_values.pkl’) unmeth = pd.read_pickle(‘unmeth_values.pkl’)` THIS will run the fastest.

**(meth and unmeth and poobah)** if `poobah=None` (default): Does nothing if `poobah=False`: suppresses this color if `poobah=dataframe`: color-codes samples according to percent probe failure range,

but only if you pass in meth and unmeth dataframes too, not data\_containers object.

if `poobah=True`: looks for `poobah_values.pkl` in the path provided.

**cutoff\_line:** True will draw the line; False omits it. **bad\_sample\_cutoff** (default 11.5): set the cutoff for determining good vs bad samples, based on signal intensities of meth and unmeth fluorescence channels. 10.5 was borrowed from minfi’s internal defaults. **noob**: use noob-corrected meth/unmeth values **verbose**: additional messages **plot**: if True (default), shows a plot. if False, this function returns the median values per sample of meth and unmeth probes. **return\_fig** (False default), if True, and plot is True, returns a figure object instead of showing plot. **compare**: if the processed data contains both noob and uncorrected values, it will plot both in different colors **palette**: if using poobah to color code, you can specify a Seaborn palette to use.

this will draw a diagonal line on plots

A dictionary of data about good/bad samples based on signal intensity

**TODO:** doesn't return both types of data if using compare and not plotting doesn't give good error message for compare

#### 4.6.8 methylcheck.plot\_M\_vs\_U

```
methylcheck.plot_M_vs_U(data_containers_or_path=None, meth=None, unmeth=None,
                        poobah=None, noob=True, silent=False, verbose=False, plot=True,
                        compare=False, return_fig=False, palette=None, cutoff_line=True)
plot methylated vs unmethylated probe intensities
```

**PATH to csv files processed using methylprep** these have “noob\_meth” and “noob\_unmeth” columns per sample file this function can use. if you want it to processed data uncorrected data. (If there is a poobah\_values.pkl file in this PATH, it will use the file to color code points)

```
data_containers = run_pipeline(data_dir = 'somepath',
                               save_uncorrected=True, sample_sheet_filepath='samplesheet.csv')
```

you can also recreate the list of datacontainers using methylcheck.load(<filepath>,'meth')

**(meth and unmeth)** if you chose *process -all* you can load the raw intensities like this, and pass them in: `meth = pd.read_pickle('meth_values.pkl')` `unmeth = pd.read_pickle('unmeth_values.pkl')` THIS will run the fastest.

**poobah** filepath: You may supply the file path to the p-value detection dataframe. If supplied, it will color code points on the plot. False: set poobah to False to suppress this coloring. None (default): if there is a poobah\_values.pkl file in your path, it will use it.

**optional params:** noob: use noob-corrected meth/unmeth values verbose: additional messages plot: if True (default), shows a plot. if False, this function returns the median values per sample of meth and unmeth probes. return\_fig: (False default), if True (and plot is true), returns the figure object instead of showing it. compare:

if the processed data contains both noob and uncorrected values, it will plot both in different colors the compare option will not work with using the 'meth' and 'unmeth' inputs, only with path or data\_containers.

**cutoff\_line: True will draw a diagonal line on plots.** the cutoff line is based on the X-Y scale of the plot, which depends on the range of intensity values in your data set.

**TODO:** doesn't return both types of data if using compare and not plotting doesn't give good error message for compare

#### 4.6.9 methylcheck.plot\_controls

```
methylcheck.plot_controls(path=None, subset='all', return_fig=False)
internal array QC controls (available with the -save_control or -all methylprep process option)
```

**path** can either be a path to the file, or a path to the folder containing a file called 'control\_probes.pkl', or it can be the dictionary of control dataframes in *control\_probes.pkl*.

**subset** ('staining' | 'negative' | 'hybridization' | 'extension' | 'bisulfite' | 'non-polymorphic' | 'target-removal' | 'specificity' | 'all');

'all' will plot every control function (default)

**return\_fig** (**False**) if True, returns a list of matplotlib.pyplot figure objects INSTEAD of showing then. Used in QC ReportPDF.

if there are more than 30 samples, plots will not have sample names on x-axis.

#### 4.6.10 methylcheck.plot\_beta\_by\_type

`methylcheck.plot_beta_by_type(beta_df, probe_type='all', return_fig=False, silent=False, on_lambda=False)`  
compare betas for type I and II probes – (inspired by the plotBetasByType() function)

Plot the overall density distribution of beta values and the density distributions of the Infinium I or II probe types 1 distribution plot; user defines type (I or II infinium)

Doesn't work with 27k arrays because they are all of the same type, Infinium Type I.

**options:** return\_fig: (default False) if True, returns a list of figure objects instead of showing plots.

#### 4.6.11 methylcheck.probes

#### 4.6.12 methylcheck.list\_problem\_probes

`methylcheck.list_problem_probes(array, criteria=None, custom_list=None)`

Function to create a list of probes to exclude from downstream processes.

By default, all probes that have been noted in the literature to have polymorphisms, cross-hybridization, repeat sequence elements and base color changes are included in the DEFAULT exclusion list.

- You can customize the exclusion list by passing in either publication shortnames or criteria into the function.
- you can combine pubs and reasons into the same list of exclusion criteria.
- if a publication doesn't match your array type, it will raise an error and tell you.

Including any of these labels in pubs (publications) or criteria (described below) will result in these probes NOT being included in the final exclusion list.

User also has ability to add custom list of probes to include in final returned list.

Parameters:

**array:** **string** name for type of array used 'IlluminaHumanMethylationEPIC', 'IlluminaHumanMethylation450k' This shorthand names are also okay:

```
{ 'EPIC', 'EPIC+', '450k', '27k', 'MOUSE' }
```

**criteria:** **list** List of the publications to use when excluding probes. If the array is 450K the publications may include: 'Chen2013' 'Price2013' 'Zhou2016' 'Naeem2014' 'DacaRoszak2015'

If the array is EPIC the publications may include: 'Zhou2016' 'McCartney2016'

If array is EPIC or EPIC+, specifying 'illumina' will remove 998 probes the manufacturer has recommended be excluded. The defects only affected a small number of EPIC arrays produced.

If no publication list is specified, probes from all publications will be added to the exclusion list. If more than one publication is specified, all probes from all publications in the list will be added to the exclusion list.

**criteria: lists** List of the criteria to use when excluding probes. List may contain the following exculsion criteria: “Polymorphism”

‘CrossHybridization’ ‘BaseColorChange’ ‘RepeatSequenceElements’ ‘illumina’

If no criteria list is specified, all critera will be excluded. If more than one criteria is specified, all probes meeting any of the listed criteria will be added to the exclusion list.

**custom\_list: list, default None** User-provided list of probes to be excluded. These probe names have to match the probe names in your data exactly.

Returns:

**probe\_exclusion\_list: list** List containing probe identifiers to be excluded

**or probe\_exclusion\_dataframe: dataframe** DataFrame containing probe names as index and reason | paper\_reference as columns

If you supply no criteria (default), then maximum filtering occurs:

- EPIC will have 389050 probes removed
- 450k arrays will have 341057 probes removed

Reason lists for 450k and probes removed:

- Daca-Roszak\_etal\_2015 (96427)
- Chen\_etal\_2013 (445389)
- Naeem\_etal\_2014 (146590)
- Price\_etal\_2013 (284476)
- Zhou\_etal\_2016 (184302)
- Polymorphism (796290)
- CrossHybridization (211330)
- BaseColorChange (359)
- RepeatSequenceElements (149205)

Reason lists for epic and probes removed:

- McCartney\_etal\_2016 (384537)
- Zhou\_etal\_2016 (293870)
- CrossHybridization (173793)
- Polymorphism (504208)
- BaseColorChange (406)

### 4.6.13 methylcheck.exclude\_probes

`methylcheck.exclude_probes(df, probe_list)`

Exclude probes from a dataframe of sample beta values. Use `list_problem_probes()` to obtain a list of probes (or pass in the names of ‘Criteria’ from problem probes), then pass that in as a `probe_list` along with the dataframe of beta values (array)



Resolves a problem whereby probe lists have basic names, but samples have additional meta data added. Example:

```
probe list ['cg24168924', 'cg15886294', 'cg05943251', 'cg05579622', 'cg01797553', 'cg14885690',
'cg12490816', 'cg02631583', 'cg17361593', 'cg15000031', 'cg21515494', 'cg17219246', 'cg10838001',
'cg13913475', 'cg00492169', 'cg20352786', 'cg05932698', 'cg06736139', 'cg08333283', 'cg10010298',
'cg25984048', 'cg27287823', 'cg19269713', 'cg12456833', 'cg26161708', 'cg04984052', 'cg00033806',
'cg23255774', 'cg10717379', 'cg00880984', 'cg01818617', 'cg18563133', 'cg15895341', 'cg08155050',
'cg06820286', 'cg04325909', 'cg15094920', 'cg08037129', 'cg11161730', 'cg06044537', 'cg11936560',
'cg12404870', 'cg12670496', 'cg01473643', 'cg08605930', 'cg16553354', 'cg22175254', 'cg22966295',
'cg07346931', 'cg06234741']
```

sample probe names

```
Index(['cg00000029_II_F_C_rep1_EPIC', 'cg00000103_II_F_C_rep1_EPIC',
'cg00000109_II_F_C_rep1_EPIC', 'cg00000155_II_F_C_rep1_EPIC',
'cg00000158_II_F_C_rep1_EPIC', 'cg00000165_II_R_C_rep1_EPIC',
'cg00000221_II_R_C_rep1_EPIC', 'cg00000236_II_R_C_rep1_EPIC', ...
'ch.9.98957343R_II_R_O_rep1_EPIC', 'ch.9.98959675F_II_F_O_rep1_EPIC',
'ch.9.98989607R_II_R_O_rep1_EPIC', 'ch.9.991104F_II_F_O_rep1_EPIC']
```

This chops off anything after the first underscore, and compares with probe\_list to see if percent match increases. It then drops probes from array that match probe\_list, at least partially.

ADDED: checking whether array.index is string or int type. Regardless, this should work and not alter the original index. ADDED v0.6.4: pass in a string like 'illumina' or 'McCartney2016' and it will fetch the list for you.

ref: <https://bioconductor.org/packages/devel/bioc/vignettes/sesame/inst/doc/sesame.html#howwhy-probes-are-masked> SESAME probe exclusion lists were pulled using these R commands:

```
EPIC_Zhou = sesameDataGet('EPIC.probeInfo')$mask # 104454 probes
HM450_Zhou <- sesameDataGet('HM450.probeInfo')$mask # 65144 probes
```

#### 4.6.14 methycheck.exclude\_sex\_control\_probes

`methycheck.exclude_sex_control_probes(df, array, no_sex=True, no_control=True, verbose=False)`

Exclude probes from an array, and return a filtered array.

df: dataframe of beta values or m-values. array: type of array used.

```
{'27k', '450k', 'EPIC', 'EPICPLUS', 'MOUSE'} or {'IlluminaHumanMethylation27k', 'IlluminaHumanMethylation450k', 'IlluminaHumanMethylationEPIC', 'mouse'} or
{'27k', '450k', 'epic', 'epic+', 'mouse'}
```

**no\_sex: bool** (default True) if True, will remove all probes that target X and Y chromosome locations, as they are sex specific – and lead to multiple clusters when trying to detect and remove outliers (noisy data).

**no\_control: bool** (default True) if True, removes Illumina's internal control probes.

**verbose: bool** (default False) reports out on number of probes removed.

a dataframe with samples removed.



#### 4.6.15 methylcheck.drop\_nan\_probes

`methylcheck.drop_nan_probes(df, silent=False, verbose=False)`  
accounts for df shape (probes in rows or cols) so dropna() will work.

the method used inside MDS may be faster, but doesn't tell you which probes were dropped.

#### 4.6.16 methylcheck.samples

#### 4.6.17 methylcheck.sample\_plot

`methylcheck.sample_plot(df, **kwargs)`

A more intuitive alias of `beta_density_plot()`, since not all values are beta distributions. Note: This changes the “`beta_density_plot()`” defaults to show a reduced, faster version of probe data, sampling 10% of probes present for 10-fold faster processing time.

#### 4.6.18 methylcheck.beta\_density\_plot

`methylcheck.beta_density_plot(df, verbose=False, save=False, silent=False, reduce=0.1, plot_title=None, ymax=None, return_fig=False, full_range=False, highlight_samples=None, figsize=(12, 9), show_labels=None, filename='beta.png')`

Returns a plot of beta values for each sample in a batch of samples as a separate line. Y-axis values is an arbitrary scale, similar to a histogram of probes that have a given beta value. X-axis values are beta values (0 to 1) for a single samples

##### Input (df):

- a dataframe with probes in rows and sample\_ids in columns.
- to get this formatted import, use `methylprep consolidate_values_for_sheet()`, as this will return a matrix of beta-values for a batch of samples (by default).

**Returns:** None (but if `return_fig` is True, returns the figure object instead of showing plot)

##### Parameters:

**verbose:** display extra messages

**save:** if True, saves a copy of the plot as a png file.

**silent:** if True, eliminates all messages (useful for automation and scripting)

**reduce:** when working with datasets and you don't need publication quality “exact” plots, supply a float between 0 and 1 to sample the probe data for plotting. We recommend 0.1, which plots 10% of the 450k or 860k probes, and doesn't distort the distribution much. Values below 0.001 (860 probes out of 860k) will show some sampling distortion. Using 0.1 will speed up plotting 10-fold of large batches.

**ymax (None):** If defined, upper limit of plot will not exceed this value. But it y-range can be smaller if values are less than this range.

**full\_range: (False)** if True, x-axis will be auto-scaled, instead of fixed in the 0-to-1.0 range.

**return\_fig: (False)** if True, returns figure object instead of showing plot.

**highlight\_samples:** a string or list of df col-names that, if provided, will highlight sample(s) in blue and bold in plot returned. all other samples in df will be grayed out. Useful for QC reports.

**figsize:** tuple of width, height, with 12,9 being default if omitted.

`show_labels`: By default, sample names appear in a legend if there are <30 samples. Otherwise, omitted. Use this to force legend on or off.

**Note:** if the `sample_ids` in `df.index` are not unique, it will make them so for the purpose of plotting.

#### 4.6.19 methylcheck.mean\_beta\_plot

`methylcheck.mean_beta_plot(df, verbose=False, save=False, silent=False)`

Returns a plot of the average beta values for all probes in a batch of samples.

**Input (df):**

- a dataframe with probes in rows and `sample_ids` in columns.
- to get this formatted import, use `methylprep consolidate_values_for_sheet()`,

as this will return a matrix of beta-values for a batch of samples (by default).

#### 4.6.20 methylcheck.mean\_beta\_compare

`methylcheck.mean_beta_compare(df1, df2, save=False, verbose=False, silent=False)`

Use this function to compare two dataframes, pre-vs-post filtering and removal of outliers. args:

the first argument (`df1`) is the “pre” dataframe of samples the second argument (`df2`) is the “post” dataframe of samples

**kwargs:** `verbose`: additional output `silent`: suppresses figure, so no output unless `save==True` too.

#### 4.6.21 methylcheck.beta\_mds\_plot

`methylcheck.beta_mds_plot(df, filter_stdev=1.5, verbose=False, save=False, silent=False, multi_params={'draw_box': True}, plot_removed=False, nafill='quick', poobah=None, palette=None, labels=None, extend_poobah_range=True, plot=True)`

Performs multidimensional scaling on a dataframe of samples

**Arguments**

**df:** dataframe of beta values for a batch of samples (rows are probes; cols are samples)

**filter\_stdev:** a value (unit: standard deviations) between 0 and 3 (typically) that represents the fraction of samples to include, based on the standard deviation of this batch of samples. So using the default value of 1.5 means that all samples whose MDS-transformed beta sort\_values are within +/- 1.5 standard deviations of the average beta are retained in the data returned.

**plot\_removed:** if True, displays a plot of samples’ beta-distributions that were removed by MDS filtering. ignored if `silent=True`.

**nafill:** (‘quick’ | ‘impute’) by default, most samples will contain missing values where probes failed the signal-noise detection in methylprep. By default, it will use the fastest method of filling in samples from adjacent sample’s probe values with the ‘quick’ method. Or, if you want it to use the average value for all samples for each probe, use ‘impute’, which will be much slower.

**poobah:** path to poobah\_values.pkl file. Default is None. If supplied, this will color code dots according to percent of failed probes for each sample as a second dimension of QC on the plot. Does not filter or affect the output dataframe returned.

**palette:** Optional - Specify a matplotlib/seaborn palette name, such as 'CMRmap\_r', 'cool-warm', or 'nipy\_spectral'. Default is 'twilight'.

**labels:** pass in a dictionary with sample names found in `df` columns and a (number or string) representing the groups to assign samples to. Use this to color-code the samples against a known classification scheme, such as cell type, and observe whether the MDS clustering pattern aligns with this input parameter. This feature is not compatible with `poobah` or `multi_params`.

**extend\_poobah\_range:** True means 7 colors appear covering 0-30%. False means 5 colors and 0-20%. Default is True.

**multi\_params:** is a dict, passed into this function from a multi-compare-MDS wrapper function, containing: {return\_plot\_obj=True, fig=None, ax=None, draw\_box=False, xy\_lim=None, color\_num=0, PSF=1.2 – plot scale factor (margin beyond points to display)}

#### Options

**verbose:** If True, provides additional messages

**silent:**

- if running from command line in an automated process, you can run in *silent* mode to suppress any user interaction.
- In this case, whatever *filter\_stdev* you assign is the final value, and a file will be processed with that param.

**plot: (default True)**

- plot is False, this suppresses plots (images) from being generated and shown on screen.
- .png files are still saved if `save == True`.

#### Returns

Returns a filtered dataframe. If `return_plot_obj` is True, it returns the plot, for making overlays in `methylyze`.

#### Requires

pandas, numpy, pyplot, sklearn.manifold.MDS

#### Notes

this will remove probes from ALL samples in batch from consideration if any samples contain NaN (missing values) for that probe.

## 4.6.22 methylcheck.combine\_mds

`methylcheck.combine_mds(*args, **kwargs)`

To combine (or segment) datasets for multidimensional scaling analysis

Use this function on multiple dataframes to combine datasets, or to visualize parts of the same dataset in separate colors. It is a wrapper of `methylcheck.beta_mds_plot()` and applies multidimensional scaling to cluster similar samples based on patterns in probe values, as well as identify possible outlier samples (and exclude them).

- combine datasets,
- run MDS,
- see how each dataset (or subset) overlaps with the others on a plot,
- exclude outlier samples based on a composite cutoff box (the average bounds of the component data sets)
- calculate the percent of data excluded from the group

- **\*args:**
  - pass in any number of pandas dataframes, and it will combine them into one mds plot.
  - alternatively, you may pass in a list of filepaths as strings, and it will attempt to load these files as pickles.

but they must be pickles of pandas dataframes containing beta values or m-values
- **silent: (default False)** (automated processing mode) if True, suppresses most information and avoids prompting user for anything. silent mode processes data but doesn't show the plot.
- **save: (default False)** if True, saves the plot png to disk.
- **verbose: (default False)** if True, prints extra debug information to screen or logger.
- **filter\_stdev:** how broadly should you retain samples? units are standard deviations, defaults to 1.5 STDEV. if you increase this number, fewer outlier samples will be removed.
- returns a dataframe of transformed samples

#### 4.6.23 methycheck.cumulative\_sum\_beta\_distribution

`methycheck.cumulative_sum_beta_distribution(df, cutoff=0.7, verbose=False, save=False, silent=False)`

Attempts to filter outlier samples based on the cumulative area under the curve exceeding a reasonable value (cutoff). This method only works on poor quality samples that are better identified using ControlReporter summary (CLI: 'methycheck controls')

**Inputs:** DataFrame – wide format (probes in columns, samples in rows) cutoff (default 0.7) silent – suppresses figure, so just returns transformed data if False. if save==True: saves figure to disk.

**Returns:** dataframe with subjects removed that exceed cutoff value.

#### 4.6.24 methycheck.predict

#### 4.6.25 methycheck.get\_sex

`methycheck.get_sex(data_source, array_type=None, verbose=False, plot=False, save=False, on_lambda=False, median_cutoff=-2, include_probe_failure_percent=True, poobah_cutoff=20, custom_label=None, return_fig=False, return_labels=False)`

This will calculate and predict the sex of each sample.

**the “data\_source” can be any one of:** path – to a folder with csv data that contains processed sample data path – to a folder with the 'meth\_values.pkl' and 'unmeth\_values.pkl' dataframes path – to a folder also containing samplesheet.pkl and poobah\_values.pkl, if you want to compare predicted sex with actual sex. data\_containers – object created from methylprep.run\_pipeline() or methycheck.load(path, 'meth') tuple of (meth, unmeth) dataframes

**array\_type (string)** enum: {'27k', '450k', 'epic', 'epic+', 'mouse'} if not specified, it will load the data from data\_source and determine the array for you.

**median\_cutoff (default is -2)** the minimum difference in the medians of X and Y probe copy numbers to assign male or female (copied from the minfi sex predict function)

**include\_probe\_failure\_percent:** True: includes poobah percent per sample as column in the output table and on the plot. Note: you must supply a 'path' as data\_source to include poobah in plots.

**poobah\_cutoff** The maximum percent of sample probes that can fail before the sample fails. Default is 20 (percent) Has no effect if *include\_probe\_failure\_percent* is False.

**plot** True: creates a plot, with option to *save* as image or *return\_fig*.

**save** True: saves the plot, if plot is True

**return\_fig** If True, returns a pyplot figure instead of a dataframe. Default is False. Note: *return\_fig* will not show a plot on screen.

**return\_labels: (requires plot == True)** When using *poobah\_cutoff*, the figure only includes A-Z, 1...N labels on samples on plot to make it easier to read. So to get what *sample\_ids* these labels correspond to, you can rerun the function with *return\_labels=True* and it will skip plotting and just return a dictionary with *sample\_ids* and these labels, to embed in a PDF report if you like.

**custom\_label:** Option to provide a dictionary with keys as *sample\_ids* and values as labels to apply to samples. e.g. add more data about samples to the multi-dimensional QC plot

while providing a filepath is the easiest way, you can also pass in a *data\_containers* object, a list of *data\_containers* containing raw *meth/unmeth* values, instead. This object is produced by *methylprep.run\_pipeline*, or by using *methylcheck.load(filepath, format='meth')* and lets you customize the import if your files were not prepared using *methylprep* (non-standard CSV columns, for example)

If a *poobah\_values.pkl* file can be found in path, the dataframe returned will also include percent of probes for X and Y chromosomes that failed quality control, and warn the user if any did. This feature won't work if a *containers* object or tuple of dataframes is passed in, instead of a path.

Note: ~90% of Y probes should fail if the sample is female. That chromosome is missing.

## 4.6.26 methylcheck.assign

*methylcheck.assign(df, two\_pass=False)*

**Manually and interactively assign each sample to a group, based on beta-value distribution shape.** This function is not documented or supported anymore.

**how:** sorts samples by the position of their peak intensity in beta dist, then lets the human assign a number to each. assumes that peaks that differ by > 0.1 of the 0 to 1 range are different clusters. fills in expected cluster #.

**options:** if *two\_pass==True*: it lets user go through every cluster a second time and split larger clusters further. sometimes a cluster isn't obviously two groups until it is separated from the larger dataset.

## 4.6.27 Loading Data

*methylcheck.load\_processed.load(filepath='.', format='beta\_value', file\_stem="", verbose=False, silent=False, column\_names=None, no\_poobah=False, pval\_cutoff=0.05, no\_filter=True)*

Methylsuite's all-purpose data loading function.

When *methylprep* processes large datasets, you use the 'batch\_size' option to keep memory and file size more manageable. Use the *load* helper function to quickly load and combine all of those parts into a single data frame of beta-values or m-values.

Doing this with *pandas* is about 8 times slower than using *numpy* in the intermediate step.

If no arguments are supplied, it will load all files in current directory that have a 'beta\_values\_X.pkl' pattern.

**Arguments:**

**filepath:** Where to look for all the pickle files of processed data.

**format:** ('beta\_value', 'm\_value', 'meth', 'meth\_df', 'noob\_df', 'beta\_csv', 'sesame') This also allows processed.csv file data to be loaded. If you need meth and unmeth values, choose 'meth' and it will return a data\_containers object with the 'meth' and 'unmeth' values, exactly like the data\_containers object returned by methylprep.run\_pipeline.

If you choose 'meth\_df' or 'noob\_df' it will load the pickled meth and unmeth dataframes from the folder specified.

**column\_names:** if your csv files contain column names that differ from those expected, you can specify them as a list of strings by default it looks for ['noob\_meth', 'noob\_unmeth'] or ['meth', 'unmeth'] or ['beta\_value'] or ['m\_value'] Note: if you csv data has probe names in a column that is not the FIRST column, or is not named "IlmnID", you should specify it with column\_names and put it first in the list, like ['illumina\_id', 'noob\_meth', 'noob\_umeth'].

**no\_poobah:** if loading from CSVs, and there is a column for probe p-values (the poobah\_pval column), the default is to filter out probes that fail the  $p < 0.05$  cutoff. if you specify 'no\_poobah'=True, it will load everything, regardless of p-values.

**pval\_cutoff:** if applying poobah (pvalue probe detection based on poor signal to noise) this specifies the threshold for cutoff (0.05 by default)

**no\_filter: (default = True)** if False, removes probes that illumina, the manufacturer, claimed are sketchy in 2019 for a select list of newer EPIC Sentrix\_IDs. only affects 'beta\_value' and 'm\_value' output; no effect on meth/unmeth raw/NOOB intensity values returned.

**file\_stem: (string)** Older versions (pre v1.3.0) of methylprep processed with batch\_size created a bunch of generically named files, such as 'beta\_values\_1.pkl', 'beta\_values\_2.pkl', 'beta\_values\_3.pkl', and so on. IF you rename these or provide a custom name during processing, provide that name here to load them all. (i.e. if your pickle file is called 'GSE150999\_beta\_values\_X.pkl', then your file\_stem is 'GSE150999\_')

**verbose:** outputs more processing messages.

**silent:** suppresses all processing messages, even warnings.

#### Use cases and format:

**format = beta\_value:** you have beta\_values.pkl file in the path specified and want a dataframe returned or you have a bunch of beta\_values\_1.pkl files in the path and want them merged and returned as one dataframe (when using 'batch\_size' option in methylprep.run\_pipeline() you'll get multiple files saved)

**format = m\_value:** you have m\_values.pkl file in the path specified and want a dataframe returned or you have a bunch of m\_values\_1.pkl files in the path and want them merged and returned as one dataframe

**format = meth: (data\_containers)** you have processed CSV files in the path specified and want a data\_container returned

**format = meth\_df: (dataframe)** you have processed CSV files in the path specified and want a dataframe returned take the data\_containers object returned and run *methylcheck.container\_to\_pkl(containers, save=True)* function on it.

**format = noob\_df: (dataframe)** loads noob\_meth\_values.pkl and noob\_unmeth\_values.pkl and returns two dataframes in a list

**format = sesame:** for reading csvs processed using R's sesame package. It has a different format (Probe\_ID, ind\_beta, ind\_negs, ind\_poob) per sample. Only those probes that pass the p-value cutoff will be included.

**format = beta\_csv:** for reading processed.csv files from methylprep, and forcing it NOT to load from the pickled beta dataframe file, if present.

**format = poobah\_csv:** similar to beta\_csv, this pulls poobah p-values for all probes out of all processed CSV files into one dataframe. These p-values will include failed probes and probes that would be filtered by quality\_mask. 'poobah' excludes these.

**format = poobah:** reads the 'poobah\_values.pkl' file and returns a dataframe of p-values. Note failed / poor-quality probes are replaced with NaN.

---

#### Note:

**Science on p-value cutoff:** This function defaults to a p-value cutoff of 0.05, which is typical for scientific tests. There is currently no consensus on what percent of a sample's probes can fail. For example, if a sample has 860,000 probes and 5% of them fail, should you reject the whole sample from the batch? For large batch industrial scale testing, the authors assign some limit, like 5%, 10%, 20%, 30%, etc as a cutoff. And methylcheck's run\_qc() function defaults to 10 percent. But the academics we spoke to don't automatically throw out any samples. Because it depends. Cancer samples have lots of aneuploidy (an abnormal number of chromosomes in a haploid set) and lost chromosomes, so one would expect no signal for these CpG sites. So those researchers wouldn't throw out samples unless most of the sample fails. People are working on deriving a calibration curve from public GEO data as a guide, and give a frame of reference, but none exist yet. And public data rarely includes failed samples.

---



---

#### Note:

- modified this from methylprep on 2020-02-20 to allow for data\_containers to be returned as option
  - v0.6.3: added 'no\_filter' step that automatically removes probes that illumina, the manufacturer, claims are sketchy for certain Catalog IDs. (Disable this with `no_filter=True`)
- 

```
methylcheck.load_processed.load_both(filepath='.', format='beta_value', file_stem="",
                                     verbose=False, silent=False, column_names=None,
                                     rename_samples=False, sample_names='Sample_Name')
```

Creates and returns TWO objects (data and meta\_data) from the given filepath. Confirms sample names match.

Returns TWO objects (data, meta) as dataframes for analysis. If meta\_data files are found in multiple folders, it will read them all and try to match to the samples in the beta\_values pickles by sample ID.

#### Arguments:

**filepath:** Where to look for all the pickle files of processed data.

**format:** 'beta\_values', 'm\_value', or some other custom file pattern.

**file\_stem (string):** By default, methylprep process with batch\_size creates a bunch of generically named files, such as 'beta\_values\_1.pkl', 'beta\_values\_2.pkl', 'beta\_values\_3.pkl', and so on. IF you rename these or provide a custom name during processing, provide that name here. (i.e. if your pickle file is called 'GSE150999\_beta\_values\_X.pkl', then your file\_stem is '**GSE150999\_**')

**column\_names:** if your processed csv files contain column names that differ from those expected, you can specify them as a list of strings by default it looks for ['noob\_meth', 'noob\_unmeth'] or ['meth', 'unmeth'] or ['beta\_value'] or ['m\_value'] Note: if you csv data has probe names in a column that is not the FIRST column, or is not named "IlmnID", you should specify it with column\_names and put it first in the list, like ['illumina\_id', 'noob\_meth', 'noob\_umeth'].



**rename\_samples:** if your meta\_data contains a ‘Sample\_Name’ column, the returned data and meta\_data will have index and columns renamed to Sample\_Names instead of Sample\_IDs, respectively.

**sample\_name (string):** the column name to use in meta dataframe for sample names. Assumes ‘Sample\_Name’ if unspecified.

**verbose:** outputs more processing messages.

**silent:** suppresses all processing messages, even warnings.

`methyIcheck.load_processed.container_to_pkI (containers, output='betas', save=True)`  
 simple helper function to convert a list of SampleDataContainer objects to a df and pickle it.

**save (True|False)** whether to save the data to disk, in the current directory

**output (‘betas’|‘m\_value’|‘meth’|‘noob’|‘copy\_number’)** reads processed CSVs and consolidates into a single dataframe, with samples in columns and probes in rows: betas – saves ‘beta\_values.pkl’ m\_value – saves ‘m\_values.pkl’ meth – saves uncorrected ‘meth\_values.pkl’ and ‘unmeth\_values.pkl’ noob – saves ‘meth\_noob\_values.pkl’ and ‘unmeth\_noob\_values.pkl’ copy\_number – saves ‘copy\_number\_values.pkl’

this is for loading a bunch of processed csv files into containers, then into betas

```
import methyIcheck as m
files = '/Volumes/202761400007'
containers = m.load(files, 'meth')
df = m.load_processed.container_to_beta(containers)
```

`methyIcheck.read_geo_processed.read_geo (filepath, verbose=False, debug=False, as_beta=True, column_pattern=None, test_only=False, rename_probe_column=True, decimals=3)`

**Use to load preprocessed GEO data into methyIcheck. Attempts to find the sample beta/M\_values in the CSV/TXT/XLSX file and turn it into a clean dataframe, with probe ids in the index/rows. Version 3 (introduced June 2020)**

- reads a downloaded file, either in csv, xlsx, pickle, txt
- looks for /d\_RxxCxx patterned headings and an probe index
- sets index in df to probes
- sets columns to sample names
- forces probe values to be floats, if strings/mixed
- if filename has ‘intensit’ or ‘signal’ in it, this converts to betas and saves even if filename doesn’t match, if columns have Methylated in them, it will convert and save
- detect multi-line headers and adjusts dataframe columns accordingly
- returns the usable dataframe

`as_beta == True` – converts meth/unmeth into a df of sample betas. `column_pattern=None` (Sample21 | Sample\_21 | Sample 21) – some string of characters that precedes the number part of each sample in the columns of the file to be ingested.

**FIXED:** [x] handle files with .Signal\_A and .Signal\_B instead of Meth/Unmeth [x] BUG: can’t parse **matrix**... files if uses underscores instead of spaces around sample numbers, or where sampleXXX has no separator. [x] handle processed files with sample\_XX [x] returns IlmnID as index/probe column, unless ‘rename\_probe\_column’ == False [x] pass in sample\_column names from header parser so that logic is in one place

(makes the output much larger, so add kwarg to exclude this)



[x] demicals (default 3) – round all probe beta/intensity/p values returned to this number of decimal places.  
 [x] bug: can only recognize beta samples if ‘sample’ in column name, or sentrix\_id pattern matches columns.

need to expand this to handle arbitrary sample naming styles (limited to one column per sample patterns)

**TODO:** [-] BUG: meth\_unmeth\_pval works *as\_beta* but not returning full data yet [-] multiline header not working with all files yet. [-] \_family GSM123456-tbl-1.txt files not detected yet

**notes:** this makes inferences based on strings in the filename, and based on the column names.

```
methylcheck.read_geo_processed.detect_header_pattern(test, filename, re-
turn_sample_column_names=False)
```

test is a dataframe with first 100 rows of the data set, and all columns. makes all the assumptions easier to read in one place.

betas non-normalized matrix\_processed matrix\_signal series\_matrix – read\_series\_matrix(file, detect\_only=True) methylated\_signal\_intensities and unmethylated\_signal\_intensities \_family

TODO: GSM12345-tbl-1.txt type files (in \_family.tar.gz packages) are possible, but needs more work. TODO: combining two files with meth/unmeth values

- numbered samples handled differently from sample\_ids in columns
- won't detect columns with no separators in strings

## 4.6.28 ReportPDF Report Builder class

**class** methylcheck.reports.qc\_report.ReportPDF (\*\*kwargs)

ReportPDF allows you to build custom QC reports.

To use:

- First, initialize the report and pass in kwargs, like `myReport = ReportPDF(**kwargs)`
- Next, run `myReport.run_qc()` to fill it in.
- Third, you must run `myReport.pdf.close()` after `run_qc()` to save the file to disk.
- You can supply kwargs to specify which QC plots to include - and supply a list of chart names to control the order of objects in the report. - if you pass in ‘order’ in kwargs, any page you omit in the order will be omitted from the final report. - You may pass in a custom table to override one of the built-in pages.
- include ‘path’ with the path to your processed pickle files.
- include an optional ‘outpath’ for where to save the pdf report.

kwargs:

- processing params - filename - poobah\_min\_percent (e.g. at least 80% of probes must pass for sample to pass) - pval\_cutoff (e.g. set alpha at 0.05) - outpath - path - runme: Default is not to actually generate all the parts of PDF with `report.run_qc()` then `report.pdf.close()`, but setting this to True will do everything at once.
- front page text - title - author - subject - keywords
- if ‘debug=True’ is in kwargs, - then it will return a report without any parts that failed.
- tests - poobah: includes a table with each sample and percent of probes that passed p-value signal detection - gct: includes GCT scores (bisulfite conversion completeness) in poobah table - mds: performs multidimensional scaling to identify and report on sample outliers

- plots - beta\_density\_plot - M\_vs\_U (default False) - M\_vs\_U\_compare (default False) – shows the effect of all processing steps vs raw intensity - qc\_signal\_intensity - controls (A battery of probe performance plots) - probe\_types
- customizing plots - poobah\_colormap (pass in the matplotlib colormap name to override the meta\_mds default colormap)

This also overrides the default colormap used in M\_vs\_U plot.

- extend\_poobah\_range (Default: True will show 7 colors for poobah failure range on beta\_mds\_plot, max 30%; False will show only 5, max 20%)
- cutoff\_line – False to disable cutoff line on qc\_signal\_intensity and M\_vs\_U plots
- appendix\_fontsize (default 12 point) – specify an int for other fontsize

custom tables:

pass in arbitrary data using kwarg custom\_tables as list of dictionaries with this structure:

```
`python custom_tables=[ { 'title': "some title, optional", # NOTE: chart
titles must be unique! 'col_names': ["<list of strings>"], 'row_names':
["<list of strings, optional>"], 'data': ["<list of lists, with order
matching col_names>"], 'order_after': "string name of the plot this should
come after. It cannot appear first in list.", 'font_size': <can be None,
int, 'auto' (shrink to page), or 'truncate' (chop of long values to make
fit)> }, {"<...second table here...>"} ] `
```

If 'order\_after' is None, the custom table will be inserted at the beginning of the report. If there are multiple custom tables and all have 'order\_after' set to None, the first table in the list gets inserted, then the next one, sequentially, so that the last table inserted will be the first table to appear.

Pre-processing pipeline:

#### Probe-level (w/explanations of suggested exclusions)

- Links to recommended probe exclusion lists/files/papers
- Background subtraction and normalization ('noob')
- Detection p-value ('neg' vs 'oob')
- Dye-bias correction (from SeSAmE)

**Sample-level (w/explanations of suggested exclusions)** Detection p-value (% failed probes) - custom detection (% failed, of those in a user-defined-list supplied to function) MDS

#### Suggested for customer to do on their own

- Sex check
- Age check
- SNP check

**\_\_init\_\_** (\*\*kwargs)

Initialize self. See help(type(self)) for accurate signature.

**exec\_summary** ()

**QC exec summary** sample\_name/ID probe % failures probe\_failure pass auto-qc result (only if present in kwargs passed in, otherwise omitted) MDS pass signal intensity pass if any fails, fail it (so overall pass)

**table 2: meta** array type (detect from data) number of samples (from data) processing pipeline version number (passed in) date processed (passed in) avg probe failure rate percent of samples that failed any failures from 'control probes'

reqs a way to capture warnings of data-off-chart

**open\_error\_buffer** ( )

preparing a stream of log messages to add to report appendix.

**page\_of\_paragraphs** (*para\_list*, *pdf*, *line\_height='double'*, *fontsize=None*)

[https://matplotlib.org/3.1.1/api/\\_as\\_gen/matplotlib.pyplot.text.html](https://matplotlib.org/3.1.1/api/_as_gen/matplotlib.pyplot.text.html) (0,0) is lower left; (1,1) is upper right.

This version estimates the size of each paragraph and moves the origin downward accordingly. This is tricky because the anchors are lower left, not upper left.

It is ok if a paragraph contains whitespace line breaks, OR each paragraph is one long line to be wrapped here. Also - if a paragraph wraps, this accounts for it in total lines count, so everything fits on a page. The default fontsize is 12 if not specified.

**page\_of\_text** (*text*, *pdf*, *fontsize=None*)

text is a single big string of text, with whitespace for line breaks. [https://matplotlib.org/3.1.1/api/\\_as\\_gen/matplotlib.pyplot.text.html](https://matplotlib.org/3.1.1/api/_as_gen/matplotlib.pyplot.text.html) (0,0) is lower left; (1,1) is upper right

**parse\_custom\_tables** (*tables*)

tables is a list of { 'title': "some title, optional", 'col\_names': [list of strings], 'row\_names': [list of strings, optional], 'data': [list of lists, with order matching col\_names], 'order\_after': string name of the plot this should come after. It cannot appear first in list. }

**to\_table** (*list\_of\_lists*, *col\_names*, *row\_names=None*, *add\_title=""*, *font\_size='auto'*)

- embeds a table in a PDF page.
- attempts to split long tables into multiple pages.
- should warn if table is too wide to fit.
- font\_size: - auto: let matplotlib figure out the best size. This breaks if one column or value is soooooooper long. - truncate: IN FUTURE, chop off each value that is too long, to force it to fit on a page. Data loss. - INT: set the font for this table at this size (for control freaks) - None: default – use font size 12.

## 4.6.29 Run QC pipeline

Run a variety of probe and sample filters in tandem, then plot results

by specifying all of your options at once, instead of running every part of methylcheck in piacemeal fashion.

this is analogous to using the methylcheck CLI, but for notebooks/scripts

**required:**

**df: (required)**

- data as a DataFrame of beta values (or DataFrame of m\_values)
- sample names in columns and probes in rows

**parameters:**

**verbose: (True/False)** default: False – shows extra info about processing if True

**silent: (True/False)** default: False – suppresses all warnings/info

**exclude\_sex:** filters out probes on sex-chromosomes

**exclude\_control:** filters out illumina control probes

**exclude\_all:** filters out the most probes (sex-linked, control, and all sketchy-listed probes from papers)

**exclude:** (list of strings, shorthand references to papers with sketchy probes to exclude)

**If the array is 450K the publications may include:** 'Chen2013' 'Price2013'  
'Zhou2016' 'Naeem2014' 'DacaRoszak2015'

**If the array is EPIC the publications may include:** 'Zhou2016' 'McCartney2016'

**or these reasons:** 'Polymorphism' 'CrossHybridization' 'BaseColorChange'  
'RepeatSequenceElements'

**or use 'exclude\_all':** to do maximum filtering, including all of these papers' lists.

**plot:** (list of strings) ['mean\_beta\_plot', 'beta\_density\_plot', 'cumulative\_sum\_beta\_distribution', 'beta\_mds\_plot', 'all'] if 'all', then all of these plots will be generated. if omitted, no plots are created.

**save\_plots:** (True|False) default: False

**export** (True|False): default: False – will export the filtered df as a pkl file if True

**note:** this pipeline cannot also apply the array-level `methycheck.run_qc()` function because that relies on additional probe information that may not be present. Everything in this pipeline applies to a dataframe of beta or m-values for a set of samples.

**returns:** a filtered dataframe object

### 4.6.30 filtering probes

`methycheck.probes.exclude_sex_control_probes(df, array, no_sex=True, no_control=True, verbose=False)`

Exclude probes from an array, and return a filtered array.

**df:** dataframe of beta values or m-values. **array:** type of array used.

{'27k', '450k', 'EPIC', 'EPICPLUS', 'MOUSE'} or {'IlluminaHumanMethylation27k', 'IlluminaHumanMethylation450k', 'IlluminaHumanMethylationEPIC', 'mouse'} or {'27k', '450k', 'epic', 'epic+', 'mouse'}

**no\_sex:** **bool** (default True) if True, will remove all probes that target X and Y chromosome locations, as they are sex specific – and lead to multiple clusters when trying to detect and remove outliers (noisy data).

**no\_control:** **bool** (default True) if True, removes Illumina's internal control probes.

**verbose:** **bool** (default False) reports out on number of probes removed.

a dataframe with samples removed.

`methycheck.probes.exclude_probes(df, probe_list)`

Exclude probes from a dataframe of sample beta values. Use `list_problem_probes()` to obtain a list of probes (or pass in the names of 'Criteria' from problem probes), then pass that in as a `probe_list` along with the dataframe of beta values (array)

Resolves a problem whereby probe lists have basic names, but samples have additional meta data added. Example:

**probe list** ['cg24168924', 'cg15886294', 'cg05943251', 'cg05579622', 'cg01797553', 'cg14885690', 'cg12490816', 'cg02631583', 'cg17361593', 'cg15000031', 'cg21515494', 'cg17219246', 'cg10838001', 'cg13913475', 'cg00492169', 'cg20352786', 'cg05932698', 'cg06736139', 'cg08333283', 'cg10010298',

```
'cg25984048', 'cg27287823', 'cg19269713', 'cg12456833', 'cg26161708', 'cg04984052', 'cg00033806',
'cg23255774', 'cg10717379', 'cg00880984', 'cg01818617', 'cg18563133', 'cg15895341', 'cg08155050',
'cg06820286', 'cg04325909', 'cg15094920', 'cg08037129', 'cg11161730', 'cg06044537', 'cg11936560',
'cg12404870', 'cg12670496', 'cg01473643', 'cg08605930', 'cg16553354', 'cg22175254', 'cg22966295',
'cg07346931', 'cg06234741']
```

#### sample probe names

```
Index(['cg00000029_II_F_C_rep1_EPIC', 'cg00000103_II_F_C_rep1_EPIC',
'cg00000109_II_F_C_rep1_EPIC', 'cg00000155_II_F_C_rep1_EPIC',
'cg00000158_II_F_C_rep1_EPIC', 'cg00000165_II_R_C_rep1_EPIC',
'cg00000221_II_R_C_rep1_EPIC', 'cg00000236_II_R_C_rep1_EPIC', ...
'ch.9.98957343R_II_R_O_rep1_EPIC', 'ch.9.98959675F_II_F_O_rep1_EPIC',
'ch.9.98989607R_II_R_O_rep1_EPIC', 'ch.9.991104F_II_F_O_rep1_EPIC']
```

This chops off anything after the first underscore, and compares with probe\_list to see if percent match increases. It then drops probes from array that match probe\_list, at least partially.

ADDED: checking whether array.index is string or int type. Regardless, this should work and not alter the original index. ADDED v0.6.4: pass in a string like 'illumina' or 'McCartney2016' and it will fetch the list for you.

ref: <https://bioconductor.org/packages/devel/bioc/vignettes/sesame/inst/doc/sesame.html#howwhy-probes-are-masked> SESAME probe exclusion lists were pulled using these R commands:

```
EPIC_Zhou = sesameDataGet('EPIC.probeInfo')$mask # 104454 probes
HM450_Zhou <- sesameDataGet('HM450.probeInfo')$mask # 65144 probes
```

`methylcheck.probes.problem_probe_reasons` (array, criteria=None)

Returns a dataframe of probes to be excluded, based on recommendations from the literature. Mouse and 27k arrays are not supported.

**array:** string name for type of array used 'IlluminaHumanMethylationEPIC', 'IlluminaHumanMethylation450k' This shorthand names are also okay:

```
{ 'EPIC', 'EPIC+', '450k', '27k', 'MOUSE', 'mouse', 'epic', '450k' }
```

**criteria:** list List of the publications to use when excluding probes. If the array is 450K the publications may include:

```
'Chen2013' 'Price2013' 'Zhou2016' 'Naeem2014' 'DacaRoszak2015' 'Sesame' – uses the default mask imported from sesame
```

**If the array is EPIC the publications may include:** 'Zhou2016' 'McCartney2016' 'Sesame' – uses the default mask imported from sesame

**or these reasons:** 'Polymorphism' 'CrossHybridization' 'BaseColorChange' 'RepeatSequenceElements'

If no publication list is specified, probes from all publications will be added to the exclusion list. If more than one publication is specified, all probes from all publications in the list will be added to the exclusion list.

**returns:** dataframe this returns a dataframe showing how each probe in the list is categorized for exclusion (based on criteria: reasons and paper-refs). This output is not suitable for other functions that just expect a list of probe names.

`methylcheck.probes.list_problem_probes` (array, criteria=None, custom\_list=None)

Function to create a list of probes to exclude from downstream processes.

By default, all probes that have been noted in the literature to have polymorphisms, cross-hybridization, repeat sequence elements and base color changes are included in the DEFAULT exclusion list.

- You can customize the exclusion list by passing in either publication shortnames or criteria into the function.
- you can combine pubs and reasons into the same list of exclusion criteria.
- if a publication doesn't match your array type, it will raise an error and tell you.

Including any of these labels in pubs (publications) or criteria (described below) will result in these probes NOT being included in the final exclusion list.

User also has ability to add custom list of probes to include in final returned list.

Parameters:

**array: string** name for type of array used 'IlluminaHumanMethylationEPIC', 'IlluminaHumanMethylation450k' This shorthand names are also okay:

```
{ 'EPIC', 'EPIC+', '450k', '27k', 'MOUSE' }
```

**criteria: list** List of the publications to use when excluding probes. If the array is 450K the publications may include: 'Chen2013' 'Price2013' 'Zhou2016' 'Naeem2014' 'DacaRoszak2015'

If the array is EPIC the publications may include: 'Zhou2016' 'McCartney2016'

If array is EPIC or EPIC+, specifying 'illumina' will remove 998 probes the manufacturer has recommended be excluded. The defects only affected a small number of EPIC arrays produced.

If no publication list is specified, probes from all publications will be added to the exclusion list. If more than one publication is specified, all probes from all publications in the list will be added to the exclusion list.

**criteria: lists** List of the criteria to use when excluding probes. List may contain the following exculsion criteria: “Polymorphism”

‘CrossHybridization’ ‘BaseColorChange’ ‘RepeatSequenceElements’ ‘illumina’“

If no criteria list is specified, all critera will be excluded. If more than one criteria is specified, all probes meeting any of the listed criteria will be added to the exclusion list.

**custom\_list: list, default None** User-provided list of probes to be excluded. These probe names have to match the probe names in your data exactly.

Returns:

**probe\_exclusion\_list: list** List containing probe identifiers to be excluded

**or probe\_exclusion\_dataframe: dataframe** DataFrame containing probe names as index and reason | paper\_reference as columns

If you supply no criteria (default), then maximum filtering occurs:

- EPIC will have 389050 probes removed
- 450k arrays will have 341057 probes removed

Reason lists for 450k and probes removed:

- Daka-Roszak\_etal\_2015 (96427)
- Chen\_etal\_2013 (445389)
- Naeem\_etal\_2014 (146590)
- Price\_etal\_2013 (284476)

- Zhou\_etal\_2016 (184302)
- Polymorphism (796290)
- CrossHybridization (211330)
- BaseColorChange (359)
- RepeatSequenceElements (149205)

Reason lists for epic and probes removed:

- McCartney\_etal\_2016 (384537)
- Zhou\_etal\_2016 (293870)
- CrossHybridization (173793)
- Polymorphism (504208)
- BaseColorChange (406)

`methylcheck.probes.drop_nan_probes(df, silent=False, verbose=False)`  
accounts for df shape (probes in rows or cols) so dropna() will work.

the method used inside MDS may be faster, but doesn't tell you which probes were dropped.

### 4.6.31 plotting functions

`methylcheck.samples.assign(df, two_pass=False)`

**Manually and interactively assign each sample to a group, based on beta-value distribution shape.** This function is not documented or supported anymore.

**how:** sorts samples by the position of their peak intensity in beta dist, then lets the human assign a number to each. assumes that peaks that differ by > 0.1 of the 0 to 1 range are different clusters. fills in expected cluster #.

**options:** if `two_pass==True`: it lets user go through every cluster a second time and split larger clusters further. sometimes a cluster isn't obviously two groups until it is separated from the larger dataset.

`methylcheck.samples.plot_assigned_groups(df, user_defined_groups)`  
takes the 'sample: group' dictionary and plots each sub-set. returns a lookup dict of each cluster → [list of samples].

`methylcheck.samples.mean_beta_plot(df, verbose=False, save=False, silent=False)`  
Returns a plot of the average beta values for all probes in a batch of samples.

**Input (df):**

- a dataframe with probes in rows and sample\_ids in columns.
- to get this formatted import, use `methylprep consolidate_values_for_sheet()`,

as this will return a matrix of beta-values for a batch of samples (by default).

`methylcheck.samples.beta_density_plot(df, verbose=False, save=False, silent=False, reduce=0.1, plot_title=None, ymax=None, return_fig=False, full_range=False, highlight_samples=None, figsize=(12, 9), show_labels=None, filename='beta.png')`

Returns a plot of beta values for each sample in a batch of samples as a separate line. Y-axis values is an arbitrary scale, similar to a histogram of probes that have a given beta value. X-axis values are beta values (0 to 1) for a single samples

**Input (df):**

- a dataframe with probes in rows and sample\_ids in columns.
- to get this formatted import, use `methycheck.prep.consolidate_values_for_sheet()`, as this will return a matrix of beta-values for a batch of samples (by default).

**Returns:** None (but if `return_fig` is True, returns the figure object instead of showing plot)

**Parameters:**

**verbose:** display extra messages

**save:** if True, saves a copy of the plot as a png file.

**silent:** if True, eliminates all messages (useful for automation and scripting)

**reduce:** when working with datasets and you don't need publication quality "exact" plots, supply a float between 0 and 1 to sample the probe data for plotting. We recommend 0.1, which plots 10% of the 450k or 860k probes, and doesn't distort the distribution much. Values below 0.001 (860 probes out of 860k) will show some sampling distortion. Using 0.1 will speed up plotting 10-fold of large batches.

**ymax (None):** If defined, upper limit of plot will not exceed this value. But it y-range can be smaller if values are less than this range.

**full\_range: (False)** if True, x-axis will be auto-scaled, instead of fixed in the 0-to-1.0 range.

**return\_fig: (False)** if True, returns figure object instead of showing plot.

**highlight\_samples:** a string or list of df col-names that, if provided, will highlight sample(s) in blue and bold in plot returned. all other samples in df will be grayed out. Useful for QC reports.

**figsize:** tuple of width, height, with 12,9 being default if omitted.

**show\_labels:** By default, sample names appear in a legend if there are <30 samples. Otherwise, omitted. Use this to force legend on or off.

**Note:** if the sample\_ids in df.index are not unique, it will make them so for the purpose of plotting.

`methycheck.samples.sample_plot(df, **kwargs)`

A more intuitive alias of `beta_density_plot()`, since not all values are beta distributions. Note: This changes the "beta\_density\_plot()" defaults to show a reduced, faster version of probe data, sampling 10% of probes present for 10-fold faster processing time.

`methycheck.samples.cumulative_sum_beta_distribution(df, cutoff=0.7, verbose=False, save=False, silent=False)`

Attempts to filter outlier samples based on the cumulative area under the curve exceeding a reasonable value (cutoff). This method only works on poor quality samples that are better identified using ControlReporter summary (CLI: 'methycheck controls')

**Inputs:** DataFrame – wide format (probes in columns, samples in rows) cutoff (default 0.7) silent – suppresses figure, so just returns transformed data if False. if `save==True`: saves figure to disk.

**Returns:** dataframe with subjects removed that exceed cutoff value.

`methycheck.samples.beta_mds_plot(df, filter_stdev=1.5, verbose=False, save=False, silent=False, multi_params={'draw_box': True}, plot_removed=False, nafill='quick', poobah=None, palette=None, labels=None, extend_poobah_range=True, plot=True)`

Performs multidimensional scaling on a dataframe of samples

Arguments

**df:** dataframe of beta values for a batch of samples (rows are probes; cols are samples)



**filter\_stdev:** a value (unit: standard deviations) between 0 and 3 (typically) that represents the fraction of samples to include, based on the standard deviation of this batch of samples. So using the default value of 1.5 means that all samples whose MDS-transformed beta sort\_values are within +/- 1.5 standard deviations of the average beta are retained in the data returned.

**plot\_removed:** if True, displays a plot of samples' beta-distributions that were removed by MDS filtering. ignored if silent=True.

**nafill:** ('quick' | 'impute') by default, most samples will contain missing values where probes failed the signal-noise detection in methylprep. By default, it will use the fastest method of filling in samples from adjacent sample's probe values with the 'quick' method. Or, if you want it to use the average value for all samples for each probe, use 'impute', which will be much slower.

**poobah:** path to poobah\_values.pkl file. Default is None. If supplied, this will color code dots according to percent of failed probes for each sample as a second dimension of QC on the plot. Does not filter or affect the output dataframe returned.

**palette:** Optional - Specify a matplotlib/seaborn palette name, such as 'CMRmap\_r', 'cool-warm', or 'nipy\_spectral'. Default is 'twilight'.

**labels:** pass in a dictionary with sample names found in df columns and a (number or string) representing the groups to assign samples to. Use this to color-code the samples against a known classification scheme, such as cell type, and observe whether the MDS clustering pattern aligns with this input parameter. This feature is not compatible with poobah or multi\_params.

**extend\_poobah\_range:** True means 7 colors appear covering 0-30%. False means 5 colors and 0-20%. Default is True.

**multi\_params:** is a dict, passed into this function from a multi-compare-MDS wrapper function, containing: {return\_plot\_obj=True, fig=None, ax=None, draw\_box=False, xy\_lim=None, color\_num=0, PSF=1.2 - plot scale factor (margin beyond points to display)}

#### Options

**verbose:** If True, provides additional messages

**silent:**

- if running from command line in an automated process, you can run in *silent* mode to suppress any user interaction.
- In this case, whatever *filter\_stdev* you assign is the final value, and a file will be processed with that param.

**plot: (default True)**

- plot is False, this suppresses plots (images) from being generated and shown on screen.
- .png files are still saved if `save == True`.

#### Returns

Returns a filtered dataframe. If `return_plot_obj` is True, it returns the plot, for making overlays in `methylyze`.

#### Requires

pandas, numpy, pyplot, sklearn.manifold.MDS

#### Notes

this will remove probes from ALL samples in batch from consideration if any samples contain NaN (missing values) for that probe.

`methyIcheck.samples.mean_beta_compare(df1, df2, save=False, verbose=False, silent=False)`

Use this function to compare two dataframes, pre-vs-post filtering and removal of outliers. args:

the first argument (df1) is the “pre” dataframe of samples the second argument (df2) is the “post” dataframe of samples

**kwargs:** verbose: additional output silent: suppresses figure, so no output unless save==True too.

`methyIcheck.samples.combine_mds(*args, **kwargs)`

To combine (or segment) datasets for multidimensional scaling analysis

Use this function on multiple dataframes to combine datasets, or to visualize parts of the same dataset in separate colors. It is a wrapper of `methyIcheck.beta_mds_plot()` and applies multidimensional scaling to cluster similar samples based on patterns in probe values, as well as identify possible outlier samples (and exclude them).

- combine datasets,
- run MDS,
- see how each dataset (or subset) overlaps with the others on a plot,
- exclude outlier samples based on a composite cutoff box (the average bounds of the component data sets)
- calculate the percent of data excluded from the group
- **\*args:**
  - pass in any number of pandas dataframes, and it will combine them into one mds plot.
  - alternatively, you may pass in a list of filepaths as strings, and it will attempt to load these files as pickles.but they must be pickles of pandas dataframes containing beta values or m-values
- **silent: (default False)** (automated processing mode) if True, suppresses most information and avoids prompting user for anything. silent mode processes data but doesn’t show the plot.
- **save: (default False)** if True, saves the plot png to disk.
- **verbose: (default False)** if True, prints extra debug information to screen or logger.
- **filter\_stdev:** how broadly should you retain samples? units are standard deviations, defaults to 1.5 STDEV. if you increase this number, fewer outlier samples will be removed.
- returns a dataframe of transformed samples

`methyIcheck.qc_plot.run_qc(path)`

Generates all QC plots for a dataset in the path provided. if *process -all* was used to create control probes and raw values for QC, because it uses four output files:

- beta\_values.pkl
- control\_probes.pkl
- meth\_values.pkl or noob\_meth\_values.pkl
- unmeth\_values.pkl or noob\_unmeth\_values.pkl

output is all to screen, so best to use in a jupyter notebook. If you prefer output in a PDF, use ReportPDF instead.

Note: this will only look in the path folder; it doesn’t do a recursive search for matching files.

```
methylcheck.qc_plot.plot_beta_by_type(beta_df, probe_type='all', return_fig=False,
                                     silent=False, on_lambda=False)
```

compare betas for type I and II probes – (inspired by the plotBetasByType() function)

Plot the overall density distribution of beta values and the density distributions of the Infinium I or II probe types  
1 distribution plot; user defines type (I or II infinium)

Doesn't work with 27k arrays because they are all of the same type, Infinium Type I.

**options:** return\_fig: (default False) if True, returns a list of figure objects instead of showing plots.

```
methylcheck.qc_plot.qc_signal_intensity(data_containers=None, path=None,
                                       meth=None, unmeth=None, poobah=None,
                                       palette=None, noob=True, silent=False, ver-
                                       bose=False, plot=True, cutoff_line=True,
                                       bad_sample_cutoff=11.5, return_fig=False)
```

Suggests sample outliers based on methylated and unmethylated signal intensity.

**path** to csv files processed using methylprep these have “noob\_meth” and “noob\_unmeth” columns per sample file this function can use. if you want it to processed data uncorrected data.

**data\_containers** output from the methylprep.run\_pipeline() command when run in a script or notebook. you can also recreate the list of datacontainers using methylcheck.load(<filepath>,'meth')

**(meth and unmeth)** if you chose *process -all* you can load the raw intensities like this, and pass them in: meth = pd.read\_pickle('meth\_values.pkl') unmeth = pd.read\_pickle('unmeth\_values.pkl') THIS will run the fastest.

**(meth and unmeth and poobah)** if poobah=None (default): Does nothing if poobah=False: suppresses this color if poobah=dataframe: color-codes samples according to percent probe failure range,

but only if you pass in meth and unmeth dataframes too, not data\_containers object.

if poobah=True: looks for poobah\_values.pkl in the path provided.

cutoff\_line: True will draw the line; False omits it. bad\_sample\_cutoff (default 11.5): set the cutoff for determining good vs bad samples, based on signal intensities of meth and unmeth fluorescence channels. 10.5 was borrowed from minfi's internal defaults. noob: use noob-corrected meth/unmeth values verbose: additional messages plot: if True (default), shows a plot. if False, this function returns the median values per sample of meth and unmeth probes. return\_fig (False default), if True, and plot is True, returns a figure object instead of showing plot. compare: if the processed data contains both noob and uncorrected values, it will plot both in different colors palette: if using poobah to color code, you can specify a Seaborn palette to use.

this will draw a diagonal line on plots

A dictionary of data about good/bad samples based on signal intensity

**TODO:** doesn't return both types of data if using compare and not plotting doesn't give good error message for compare

```
methylcheck.qc_plot.plot_M_vs_U(data_containers_or_path=None, meth=None, unmeth=None,
                                poobah=None, noob=True, silent=False, verbose=False,
                                plot=True, compare=False, return_fig=False, palette=None,
                                cutoff_line=True)
```

plot methylated vs unmethylated probe intensities

**PATH to csv files processed using methylprep** these have “noob\_meth” and “noob\_unmeth” columns per sample file this function can use. if you want it to processed data uncorrected data. (If there is a poobah\_values.pkl file in this PATH, it will use the file to color code points)

**data\_containers = run\_pipeline(data\_dir = ‘somepath’,**

save\_uncorrected=True, sample\_sheet\_filepath=‘samplesheet.csv’)

you can also recreate the list of datacontainers using methylcheck.load(<filepath>,’meth’)

**(meth and unmeth)** if you chose *process -all* you can load the raw intensities like this, and pass them in: meth = pd.read\_pickle(‘meth\_values.pkl’) unmeth = pd.read\_pickle(‘unmeth\_values.pkl’) THIS will run the fastest.

**poobah** filepath: You may supply the file path to the p-value detection dataframe. If supplied, it will color code points on the plot. False: set poobah to False to suppress this coloring. None (default): if there is a poobah\_values.pkl file in your path, it will use it.

**optional params:** noob: use noob-corrected meth/unmeth values verbose: additional messages plot: if True (default), shows a plot. if False, this function returns the median values per sample of meth and unmeth probes. return\_fig: (False default), if True (and plot is true), returns the figure object instead of showing it. compare:

if the processed data contains both noob and uncorrected values, it will plot both in different colors the compare option will not work with using the ‘meth’ and ‘unmeth’ inputs, only with path or data\_containers.

**cutoff\_line: True will draw a diagonal line on plots.** the cutoff line is based on the X-Y scale of the plot, which depends on the range of intensity values in your data set.

**TODO:** doesn’t return both types of data if using compare and not plotting doesn’t give good error message for compare

`methyIcheck.qc_plot.plot_controls (path=None, subset=‘all’, return_fig=False)`

internal array QC controls (available with the *-save\_control* or *-all* methylprep process option)

**path** can either be a path to the file, or a path to the folder containing a file called ‘control\_probes.pkl’, or it can be the dictionary of control dataframes in *control\_probes.pkl*.

**subset** (‘staining’ | ‘negative’ | ‘hybridization’ | ‘extension’ | ‘bisulfite’ | ‘non-polymorphic’ | ‘target-removal’ | ‘specificity’ | ‘all’):

‘all’ will plot every control function (default)

**return\_fig** (False) if True, returns a list of matplotlib.pyplot figure objects INSTEAD of showing then. Used in QC ReportPDF.

if there are more than 30 samples, plots will not have sample names on x-axis.

`methyIcheck.qc_plot.bis_conversion_control (path_or_df, use_median=False, on_lambda=False, verbose=False)`

GCT score: requires path to noob\_meth or raw meth\_values.pkl; or you can pass in a meth dataframe. use\_median: not supported yet. Always uses mean of probe values

### 4.6.32 sex prediction

```
methylcheck.predict.get_sex(data_source, array_type=None, verbose=False, plot=False,
                             save=False, on_lambda=False, median_cutoff=-2, include_probe_failure_percent=True,
                             poobah_cutoff=20, custom_label=None, return_fig=False, return_labels=False)
```

This will calculate and predict the sex of each sample.

**the “data\_source” can be any one of:** path – to a folder with csv data that contains processed sample data path – to a folder with the ‘meth\_values.pkl’ and ‘unmeth\_values.pkl’ dataframes path – to a folder also containing samplesheet.pkl and poobah\_values.pkl, if you want to compare predicted sex with actual sex. data\_containers – object created from methylprep.run\_pipeline() or methylcheck.load(path, ‘meth’) tuple of (meth, unmeth) dataframes

**array\_type (string)** enum: { ‘27k’, ‘450k’, ‘epic’, ‘epic+’, ‘mouse’ } if not specified, it will load the data from data\_source and determine the array for you.

**median\_cutoff (default is -2)** the minimum difference in the medians of X and Y probe copy numbers to assign male or female (copied from the minfi sex predict function)

**include\_probe\_failure\_percent:** True: includes poobah percent per sample as column in the output table and on the plot. Note: you must supply a ‘path’ as data\_source to include poobah in plots.

**poobah\_cutoff** The maximum percent of sample probes that can fail before the sample fails. Default is 20 (percent) Has no effect if *include\_probe\_failure\_percent* is False.

**plot** True: creates a plot, with option to *save* as image or *return\_fig*.

**save** True: saves the plot, if plot is True

**return\_fig** If True, returns a pyplot figure instead of a dataframe. Default is False. Note: *return\_fig* will not show a plot on screen.

**return\_labels: (requires plot == True)** When using *poobah\_cutoff*, the figure only includes A-Z, 1...N labels on samples on plot to make it easier to read. So to get what sample\_ids these labels correspond to, you can rerun the function with *return\_labels=True* and it will skip plotting and just return a dictionary with sample\_ids and these labels, to embed in a PDF report if you like.

**custom\_label:** Option to provide a dictionary with keys as sample\_ids and values as labels to apply to samples. e.g. add more data about samples to the multi-dimensional QC plot

while providing a filepath is the easiest way, you can also pass in a data\_containers object, a list of data\_containers containing raw meth/unmeth values, instead. This object is produced by methylprep.run\_pipeline, or by using methylcheck.load(filepath, format=‘meth’) and lets you customize the import if your files were not prepared using methylprep (non-standard CSV columns, for example)

If a *poobah\_values.pkl* file can be found in path, the dataframe returned will also include percent of probes for X and Y chromosomes that failed quality control, and warn the user if any did. This feature won’t work if a containers object or tuple of dataframes is passed in, instead of a path.

Note: ~90% of Y probes should fail if the sample is female. That chromosome is missing.

```
methylcheck.predict.infer_strain(beta_df_or_filepath, manifest=None)
```

Uses SNPS to identify the mouse strain (among 36 possibilities), using an internal lookup ref table from sesame.

**argument:**

**beta\_df\_or\_filepath** note that beta\_df does not contain the snps, but you can provide access to a control\_probes.pkl file and it will load and pull the snps for analysis. Or, if you use *methylcheck.load(<path>, format=‘beta\_csv’)* the beta\_df WILL contain snps.

**manifest (default: None)** It load the mouse manifest by default. But if you want to provide a custom manifest, you can.

---

**Note:** This function calculates *Variant allele frequency (VAF)* in an intermediate step: - VAF is the percentage of sequence reads observed matching a specific DNA variant divided by the overall coverage at that locus. - VAF is a surrogate measure of the proportion of DNA molecules in the original specimen carrying the variant.

---

Possible Matching Strains:

'DBA\_1J', 'DBA\_2J', 'AKR\_J', 'C57L\_J', '129S5SvEvBrd', 'BALB\_cJ', 'C57BL\_10J', 'BTBR\_T+\_Itpr3tf\_J', 'LP\_J', 'A\_J', '129S1\_SvImJ', 'RF\_J', 'LEWES\_EiJ', 'PWK\_PhJ', 'BUB\_BnJ', 'SPRET\_EiJ', 'MOLF\_EiJ', 'NZB\_B1NJ', 'NZO\_HiLtJ', 'NZW\_LacJ', 'KK\_HiJ', '129P2\_OlaHsd', 'C3H\_HeJ', 'WSB\_EiJ', 'CBA\_J', 'C3H\_HeH', 'NOD\_ShiLtJ', 'C57BR\_cdJ', 'CAST\_EiJ', 'ZALLENDE\_EiJ', 'C58\_J', 'C57BL\_6NJ', 'ST\_bJ', 'I\_LnJ', 'SEA\_GnJ', 'FVB\_NJ'

## 4.7 Release History

### 4.7.1 v0.8.3

- get\_sex bug fixes; supports plots, returning figure, returning labels, or returning predicted sex dataframe
- add testing via github actions
- updated documentation

### 4.7.2 v0.8.2

- added support for sample sheets with the legacy Illumina [Header] ... [Data] format. This requires methylprep be installed for the controls report to run now.

### 4.7.3 v0.8.1

- .load gives clearer error when loading beta values from CSVs ('beta\_csv') if probe names are not unique, and returns a list of series for each sample when indices fail to merge (pandas.concat)
- .beta\_mds\_plot() can now suppress the interactive portion and still display plots, using silent=True and plot=True (plot is a new kwarg, and defaults to True). Previously silent mode would suppress both prompts and plot display. Change in behavior: silent mode will not disable plotting. Must also include plot=False for that.

### 4.7.4 v0.8.0

- Fixed bug in .load that requires tqdm >= 4.61.2
- Added more detailed error message on .load; it cannot load and merge two meth/unmeth dataframes with redundant probe names.

### 4.7.5 v0.7.9

- ReportPDF accepts 'poobah\_colormap' kwarg to feed in beta\_mds\_plot colormap.
- ReportPDF custom tables: You can insert your custom table on the first page by specifying 'order\_after' == None.
- beta\_mds\_plot palette can now be any matplotlib colormap name. Defaults to 'magma' if not specified. The palette is only used to color-code poobah failure rates, if the poobah file path is specified.
- beta\_mds\_plot new kwarg extend\_poobah\_range: Default (True) shows 7 colors for poobah failure rates. If False, will show only 5.

### 4.7.6 v0.7.6

- Reading IDATs loading bar didn't work correctly, showed up after loading.
- Fixed error/logging messages:
  - exclude\_sex\_control\_probes() said 916 control probes were removed, then said "it appears your sample had no control probes"
  - Erroneous message about missing values in poobah file: "color coding may be inaccurate."
  - Filtering probes info message said there were N samples when it meant probes.
  - methylprep.download.build\_composite\_dataset() Process time was negative.
- Target Removal and Staining graphs in plot\_controls() had unreadable X-axis sample names. Labels are suppressed when showing more than 30 samples.
- methylcheck.detect\_array() sometimes returned array types in wrong case. All functions expect lowercase array types now.
  - resolves exclude\_sex\_control\_probes bugs.
- run\_qc() and get\_sex() did not recognize poobah\_values.pkl on MacOS when using "~" in the filepath.
- methylcheck.problem\_probe\_reasons() lists probes matching any/all criteria when passing in no arguments, as documented
- get\_sex() understands samplesheet 'm' and 'f' when not capitalized now.
- Load\_both: always returns dataframe with probes in rows now, like .load() does.
- plot\_M\_vs\_U now loads the noob\_meth\_values.pkl files if noob=True and files are found; otherwise it uses whatever meth/unmeth data is available.
- Methylcheck.qc\_plot.qc\_signal\_intensity returns a dictionary of data about good/bad samples based on signal intensity. Previously it was only returning this if 'plot' was False.
- controls\_report() bug fixed: methylprep was producing samplesheet meta data pickles that contained Sample\_ID twice, because the GEO series\_matrix files had this data appear twice. This broke the report, but this case is caught and avoided now. controls\_report() will recognize a wider array of samplesheet filenames now; anything with 'samplesheet' or 'meta\_data' in the filename.

### 4.7.7 v0.7.5

- added 'methylcheck report' CLI option to create a ReportPDF
- updated documentation

- minor bug fixes in read\_geo()
  - qc\_plot() now handles mouse probe type differently
  - handles importing from multiple pandas versions correctly
  - read\_geo can open series\_matrix.txt files now

#### 4.7.8 v0.7.4

- fixed big where csv data\_files were not included in pypi

#### 4.7.9 v0.7.3

- Improved ReportPDF custom tables option
  - if fields are too long, it will truncate them or auto scale the font size smaller to fit on page.

#### 4.7.10 v0.7.2

- added GCT score to controls\_report() used in the ReportPDF class.
- ReportPDF changes
  - uses noob\_meth/unmeth instead of raw, uncorrected meth/unmeth values for GCT and U vs M plot
  - inverted poobah table to report percent passing (instead of failing) probes per sample
  - this changed input from ‘poobah\_max\_percent’ (default 5%) to ‘poobah\_min\_percent’, (default 80%)
  - M\_vs\_U not included by default, because redundant with qc\_signal\_intensity
  - M\_vs\_U compare=True now labels each sample and has legend, so you can see effect of NOOB+dye correction on batch
  - added poobah color-coding to MDS plot
- get\_sex improved plotting
  - will read poobah data and size sample points according to percent of failed probes
  - save plots, or return fig, and more options now

#### 4.7.11 v0.7.1

- Added a controls\_report() function that creates a spreadsheet summary of control probe performance.
- **New unit test coverage. Note that because methylprep v1.4.0 changes processing, the results will change slightly to match sesame instead of minfi, with nonlinear-dye-bias correction and infer-type-I-probe-switching.**
- changed org name from FoxoBioScience to FoxoTech



#### 4.7.12 v0.7.0

- Illumina Mouse Array Support
- Complete rewrite of documentation
- `qc_signal_intensity` and `plot_M_vs_U` have additional options, including superimposing poobah (percent probe failures per sample) on the plot coloring.
- `.load` will work on `control_probes.pkl` and `mouse_probes.pkl` files (with alt structure: dictionary of dataframe)
- `.sample_plot` uses “best” legend positioning now, because it was not fitting on screen with prev settings.

#### 4.7.13 v0.6.4

- `get_sex()` function returns a dataframe that also includes percent of X and Y probes that failed p-value-probe detection, as an indication of whether the predicted sex is reliable.
- better unit test coverage of predictions, `load`, `load_both`, and `container_to_pkl` functions
- fixed bug in `load( 'meth_df' )`

#### 4.7.14 v0.6.3

- fixed bug in `detect_array()` where it labeled EPIC+ as EPIC

#### 4.7.15 v0.6.2

- minor fixes to `load()` and `read_geo()`
- `exclude_probes()` accepts `problem_probes` criteria as alternate way to specify probes.
  - Exclude probes from a df of samples. Use `list_problem_probes()` to obtain a list of probes (or pass in the names of ‘Criteria’ from problem probes), then pass that in as a `probe_list` along with the dataframe of beta values (array)
- `load_processed` now has a `no_filter=False` option that will remove probes that failed p-value detection, if passing in `beta_values.pkl` and `poobah_values.pkl` files.
- `load()` now handles gzipped files the same way (so `.pkl.gz` or `.csv.gz` OK as file or folder inputs)
- seaborn v0.10 -> v0.11 deprecated the `distplot()` function that was used heavily. So now this employs `kdeplot()` in its place, with similar results.

#### 4.7.16 v0.6.1

- exposed more `beta_density_plot` parameters, so it can be used to make a QC plot (highlighting one or several samples within a larger batch, and graying out the others in the plot).

#### 4.7.17 v0.6.0

- improved `read_geo()` function, for downloading GEO methylation data sets and parsing `meta_data` from projects.
- changed org name from life-epigenetics to FoxoBioScience on Github.

#### 4.7.18 v0.5.9

- qc\_plot bug fixes -99

#### 4.7.19 v0.5.7

- -99 bug in negative controls fixed

#### 4.7.20 v0.5.4

- tweaking custom-tables in ReportPDF

#### 4.7.21 v0.5.2

- ReportPDF.run\_qc() supports on\_lambda, and any functions that require .methyIprep\_manifest\_files can be set to look for manifests in /tmp using on\_lambda=True

#### 4.7.22 v0.5.1

- sklearn now optional for MDS

#### 4.7.23 v0.5.0

- adds kwargs to functions for silent processing returning figure objects, and a report\_pdf class that can run QC and generate a PDF report.
- added **version**
- p-value probe detection
- hdbscan clustering functions
- more QC methods testing

#### 4.7.24 v0.4.0

- more tests, smart about df orientation, and re-organized files
- added read\_geo() for processed datafiles, and unit tests for it. Works with txt, csv, xls, xlsx, pkl files
- read\_geo() docs
- debugged filters.list\_problem\_probes:
- updated the docs to have correct spelling for refs/reasons.
- added a function that lets you see more detail on the probes and reasons/pubs criteria
- added more genome studio QC functions,
  - improved .load function (but not consolidated through methyl-suite yet)
  - function .assign() for manually categorizing samples
  - unit testing on the predict.sex function

- `get_sex()` prediction
- consolidated data loading for functions and uses fastest option



## CHAPTER 5

---

### Indices and tables

---

- `genindex`



### m

- `methyIcheck.cli`, [73](#)
- `methyIcheck.load_processed`, [87](#)
- `methyIcheck.predict`, [103](#)
- `methyIcheck.probes`, [94](#)
- `methyIcheck.qc_plot`, [100](#)
- `methyIcheck.read_geo_processed`, [90](#)
- `methyIcheck.reports.qc_report.run_pipeline`,  
[93](#)
- `methyIcheck.samples`, [97](#)





## Symbols

`__init__()` (*methylcheck.reports.qc\_report.ReportPDF*  
method), 92

## A

`assign()` (*in module methylcheck*), 87  
`assign()` (*in module methylcheck.samples*), 97

## B

`beta_density_plot()` (*in module methylcheck*), 83  
`beta_density_plot()` (*in module*  
*methylcheck.samples*), 97  
`beta_mds_plot()` (*in module methylcheck*), 84  
`beta_mds_plot()` (*in module methylcheck.samples*),  
98  
`bis_conversion_control()` (*in module*  
*methylcheck.qc\_plot*), 102

## C

`combine_mds()` (*in module methylcheck*), 85  
`combine_mds()` (*in module methylcheck.samples*),  
100  
`container_to_pkl()` (*in module*  
*methylcheck.load\_processed*), 90  
`cumulative_sum_beta_distribution()` (*in*  
*module methylcheck*), 86  
`cumulative_sum_beta_distribution()` (*in*  
*module methylcheck.samples*), 98

## D

`detect_header_pattern()` (*in module*  
*methylcheck.read\_geo\_processed*), 91  
`drop_nan_probes()` (*in module methylcheck*), 83  
`drop_nan_probes()` (*in module*  
*methylcheck.probes*), 97

## E

`exclude_probes()` (*in module methylcheck*), 81

`exclude_probes()` (*in module methylcheck.probes*),  
94

`exclude_sex_control_probes()` (*in module*  
*methylcheck*), 82

`exclude_sex_control_probes()` (*in module*  
*methylcheck.probes*), 94

`exec_summary()` (*methylcheck.reports.qc\_report.ReportPDF*  
method), 92

## G

`get_sex()` (*in module methylcheck*), 86  
`get_sex()` (*in module methylcheck.predict*), 103

## I

`infer_strain()` (*in module methylcheck.predict*),  
103

## L

`list_problem_probes()` (*in module methylcheck*),  
80  
`list_problem_probes()` (*in module*  
*methylcheck.probes*), 95  
`load()` (*in module methylcheck*), 76  
`load()` (*in module methylcheck.load\_processed*), 87  
`load_both()` (*in module methylcheck*), 77  
`load_both()` (*in module*  
*methylcheck.load\_processed*), 89

## M

`mean_beta_compare()` (*in module methylcheck*), 84  
`mean_beta_compare()` (*in module*  
*methylcheck.samples*), 99  
`mean_beta_plot()` (*in module methylcheck*), 84  
`mean_beta_plot()` (*in module*  
*methylcheck.samples*), 97  
`methylcheck.cli` (module), 73  
`methylcheck.load_processed` (module), 87  
`methylcheck.predict` (module), 86, 103  
`methylcheck.probes` (module), 80, 94

[methylcheck.qc\\_plot \(module\), 100](#)  
[methylcheck.read\\_geo\\_processed \(module\), 90](#)  
[methylcheck.reports.qc\\_report.run\\_pipeline \(module\), 93](#)  
[methylcheck.samples \(module\), 83, 97](#)

## O

[open\\_error\\_buffer\(\) \(methylcheck.reports.qc\\_report.ReportPDF method\), 93](#)

## P

[page\\_of\\_paragraphs\(\) \(methylcheck.reports.qc\\_report.ReportPDF method\), 93](#)  
[page\\_of\\_text\(\) \(methylcheck.reports.qc\\_report.ReportPDF method\), 93](#)  
[parse\\_custom\\_tables\(\) \(methylcheck.reports.qc\\_report.ReportPDF method\), 93](#)  
[plot\\_assigned\\_groups\(\) \(in module methylcheck.samples\), 97](#)  
[plot\\_beta\\_by\\_type\(\) \(in module methylcheck\), 80](#)  
[plot\\_beta\\_by\\_type\(\) \(in module methylcheck.qc\\_plot\), 100](#)  
[plot\\_controls\(\) \(in module methylcheck\), 79](#)  
[plot\\_controls\(\) \(in module methylcheck.qc\\_plot\), 102](#)  
[plot\\_M\\_vs\\_U\(\) \(in module methylcheck\), 79](#)  
[plot\\_M\\_vs\\_U\(\) \(in module methylcheck.qc\\_plot\), 101](#)  
[problem\\_probe\\_reasons\(\) \(in module methylcheck.probes\), 95](#)

## Q

[qc\\_signal\\_intensity\(\) \(in module methylcheck\), 78](#)  
[qc\\_signal\\_intensity\(\) \(in module methylcheck.qc\\_plot\), 101](#)

## R

[read\\_geo\(\) \(in module methylcheck\), 75](#)  
[read\\_geo\(\) \(in module methylcheck.read\\_geo\\_processed\), 90](#)  
[ReportPDF \(class in methylcheck.reports.qc\\_report\), 91](#)  
[run\\_pipeline\(\) \(in module methylcheck\), 74](#)  
[run\\_qc\(\) \(in module methylcheck\), 74](#)  
[run\\_qc\(\) \(in module methylcheck.qc\\_plot\), 100](#)

## S

[sample\\_plot\(\) \(in module methylcheck\), 83](#)  
[sample\\_plot\(\) \(in module methylcheck.samples\), 98](#)