methylcheck Documentation

Release 0.2

Life Epigenetics

Nov 05, 2019

Contents:

methylcheck is a Python-based package for filtering and visualizing Illumina methylation array data. The focus is on quality control.

CHAPTER 1

methylcheck Package

Methylcheck is designed to accept the output from the Methylprep package. It contains both high-level APIs for processing data from local files and low-level functionality allowing you to customize the flow of data and how it is processed.

CHAPTER 2

Installation

This package is available in PyPi. `pip install methylcheck`

CHAPTER 3

How to use it

In general, the best way to import data is to use `methylprep` and run `run_pipeline(data_folder, betas=True)`, collect the beta_values.pkl file it returns/saves to disk, and load that in a Jupyter notebook with methylcheck. From there, each data transformation is a single line of code using Panadas DataFrames. `methylcheck` will keep track of the data format/structures for you, and you can visualize the effect of each filter as you go. You can also export images of your charts for publication.

Refer to the Jupyter notebooks on readthedocs for examples of filtering probes from a batch of samples, removing outlier samples, and generating plots of data.

CHAPTER 4

Authors

Parts of this package were ported from `minfi`, a `R` package, and extended/developed by the team at Life Epigenetics, who maintains it. You can write to `info@LifeEgx.com` to give feedback, ask for help, or suggest improvements. For bugs, report issues on our github repo page.

4.1 Full example of processing samples

[1]:
```
# This illustrates the simplest, smallest batch (n=2) of methylation array sample␣
↪results using methylprep and methylcheck.
#python 3.7
import pandas as pd
import numpy as np
import seaborn as sb
import matplotlib.pyplot as plt
%matplotlib inline
```

[2]:
```
#import methylprep/methylcheck: run from the methylprep folder; validate using right␣
↪location files for testing
from pprint import pprint as pp
import sys
import os
sys.path.insert(0,"/Users/mmaxmeister/legx/methylprep") # not needed if pip installed
import methylprep
```

[10]:
```
# load sample data
os.chdir('/Users/mmaxmeister/legx/methylprep')
data_dir = 'docs/example_data/GSE69852'
datas = methylprep.run_pipeline(data_dir)

# this next line is an unnecessary step if you add --betas to command line, or␣
↪betas=True to the run_pipeline function.
# betas = methylprep.consolidate_values_for_sheet(datas)
```

(continues on next page)

(continued from previous page)

```
# NOTES:
#   the default "export" output is a CSV file for each sample
#   the default python interpreter scripted output is a list of dataframes, one per sample.
#   this transforms methylprep's output into a single dataframe of beta values, usable with methylcheck.
```

```
100%|| 2/2 [00:19<00:00,  9.93s/it]
```

[11]:
```
# load sample data -- simpler way
os.chdir('/Users/mmaxmeister/legx/methylprep')
data_dir = 'docs/example_data/GSE69852'
betas = methylprep.run_pipeline(data_dir, betas=True)
```

```
100%|| 2/2 [00:19<00:00,  9.97s/it]
```

[13]:
```
betas.head()
```

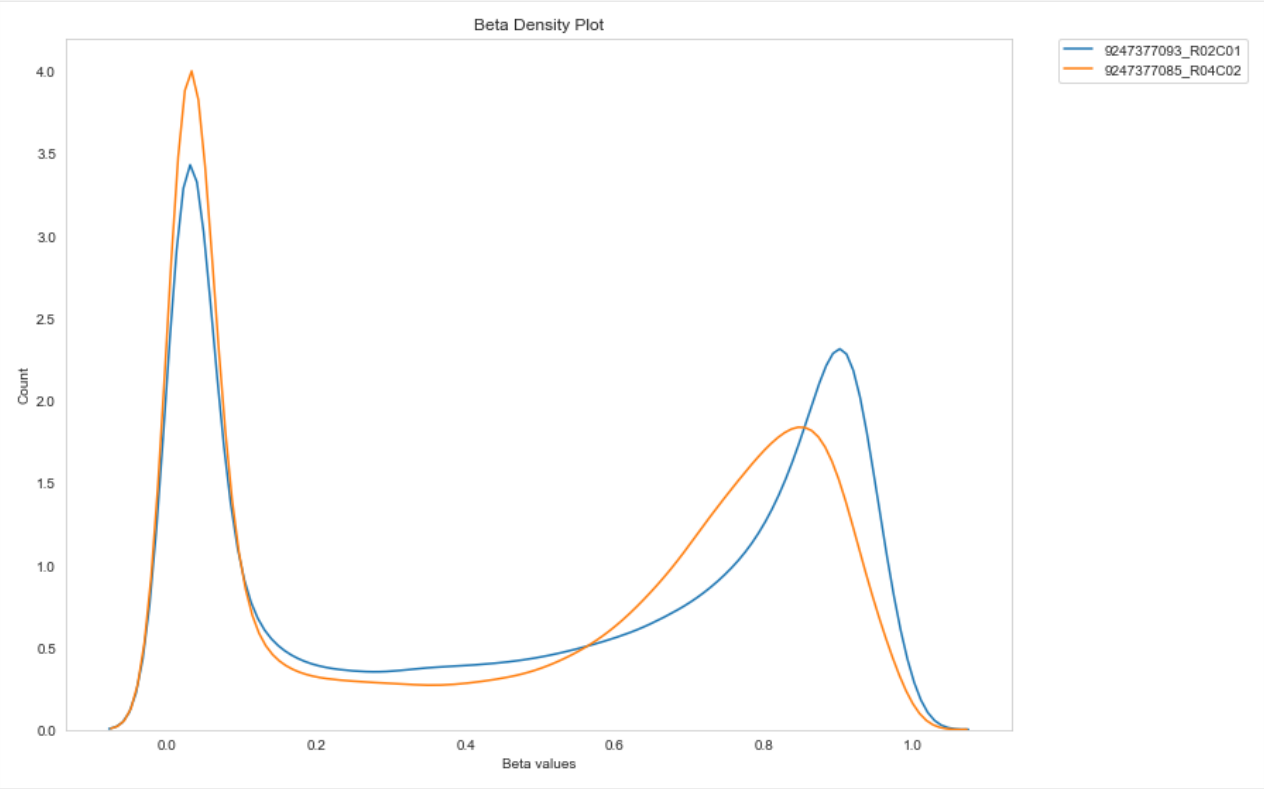
[13]:

IlmnID	9247377093_R02C01	9247377085_R04C02
cg00035864	0.236234	0.308176
cg00061679	0.427194	0.525169
cg00063477	0.929039	0.932739
cg00121626	0.481058	0.330045
cg00223952	0.044029	0.022201

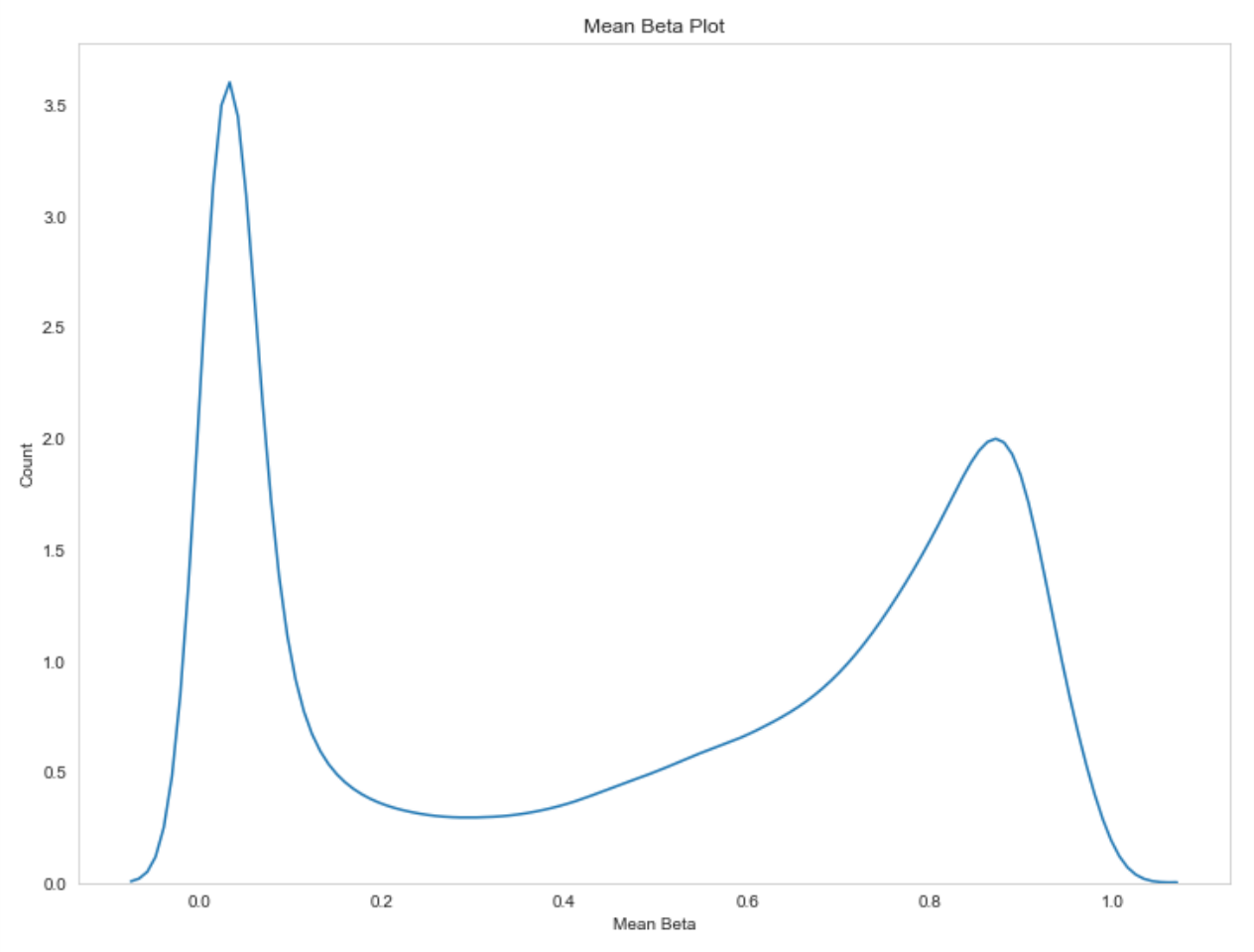
4.1.1 beta_density_plot

[19]:
```
#plot each sample separately
print(os.getcwd()) #chdir('/Users/mmaxmeister/legx/methylcheck')
import methylcheck
methylcheck.beta_density_plot(betas)
```

```
/Users/mmaxmeister/legx/methylprep
2
```



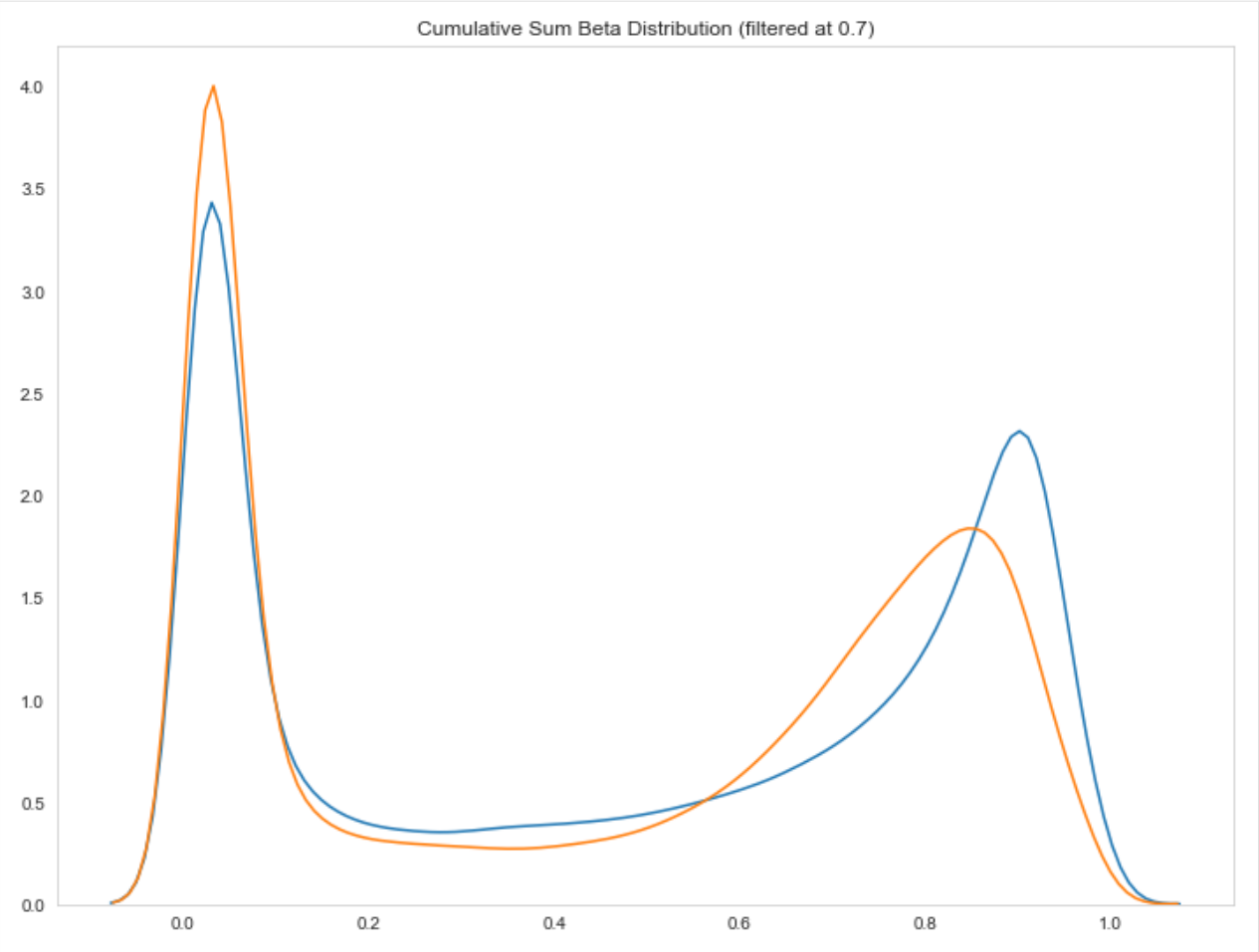
[20]:
```
# this is a mushed average of all samples into one line.
methylcheck.mean_beta_plot(betas)
```



[21]:
```
filtered_df = methylcheck.cumulative_sum_beta_distribution(betas, cutoff=0.7,␣
↪plot=True)
# use this to remove outliers, based on the cutoff value.
# this looks identical the beta_density_plot because no samples were removed.
```

```
2it [00:00, 11.76it/s]
```

```
Calculating area under curve for each sample.
```



4.2 Filtering problem probes

4.2.1 by criteria, or publication source, or sex-linked, or array-controls.

[1]:
```
%load_ext autoreload
%autoreload 2

## THE HARD WAY -- when this hasn't been pip installed yet.
#import methylprep and methylcheck -- adjust paths relative to this folder.
import os
print(os.getcwd())

import sys
methylcheck_path = os.path.abspath(os.path.join('..'))
if methylcheck_path not in sys.path:
    sys.path.insert(0,methylcheck_path)
import methylcheck

methylprep_path = os.path.abspath(os.path.join('../../methylprep'))
if methylprep_path not in sys.path:
    sys.path.append(methylprep_path)
```

(continues on next page)

(continued from previous page)

```
import methylprep

# ignore warnings for now
import warnings
warnings.filterwarnings('ignore')
print(methylcheck.__path__, methylprep.__path__)

""" ## THE EASY WAY.
import methylcheck
import methylprep
dir()
"""
```

```
/Users/mmaxmeister/legx/methylcheck/docs
['/Users/mmaxmeister/legx/methylcheck/methylcheck'] ['/Users/mmaxmeister/legx/
↪methylprep/methylprep']
```

[1]:
```
' ## THE EASY WAY.\nimport methylcheck\nimport methylprep\ndir()\n'
```

4.3 All available probe exclusion lists

[2]:
```
criteria = ['Chen2013', 'Price2013', 'Naeem2014', 'DacaRoszak2015',
            'Polymorphism', 'CrossHybridization', 'BaseColorChange',
↪'RepeatSequenceElements']
EPIC_criteria = ['McCartney2016', 'Zhou2016', 'Polymorphism', 'CrossHybridization',
↪'BaseColorChange', 'RepeatSequenceElements']

print('450k')
for crit in criteria:
    print(crit, len(methylcheck.list_problem_probes('450k', [crit])))
print('EPIC')
for crit in EPIC_criteria:
    print(crit, len(methylcheck.list_problem_probes('EPIC', [crit])))
```

```
450k
Chen2013 265410
Price2013 213246
Naeem2014 128695
DacaRoszak2015 89678
Polymorphism 289952
CrossHybridization 92524
BaseColorChange 359
RepeatSequenceElements 96631
EPIC
McCartney2016 326267
Zhou2016 178671
Polymorphism 346033
CrossHybridization 108172
BaseColorChange 406
RepeatSequenceElements 0
```

[27]:
```
# read in the sample sheet for the experiment
baseDir = "example_data/GSE69852/"
```

(continues on next page)

(continued from previous page)

```
# generate a dataframe of beta values for these samples with pipeline.
df = methylprep.run_pipeline(baseDir, betas=True)
```

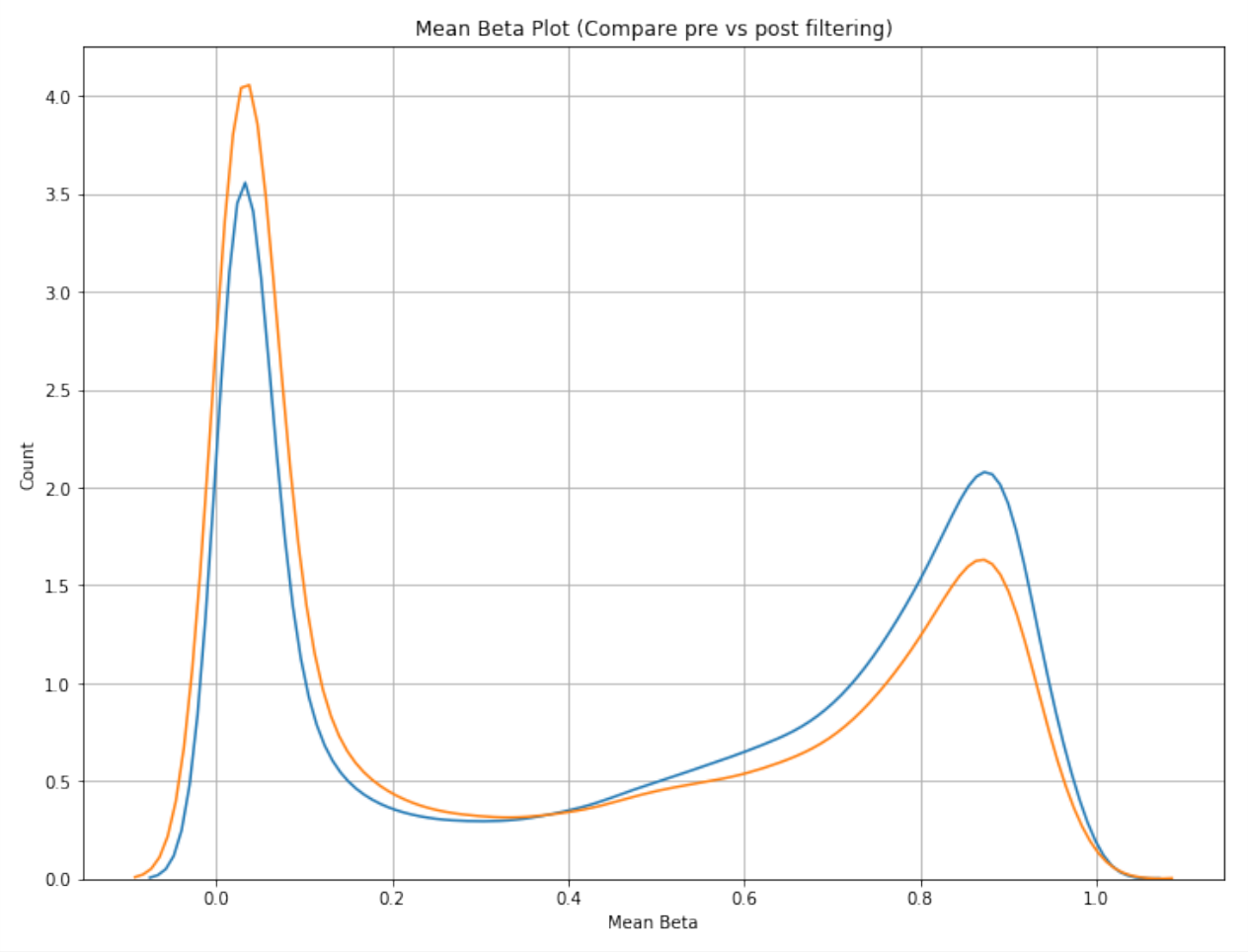
```
100%|| 6/6 [00:59<00:00,  9.84s/it]
```

[28]:
```
#import importlib
#importlib.reload(methylcheck)
df.head()
```

[28]:
```
            9247377093_R02C01  9247377093_R03C01  9247377093_R06C02  \
IlmnID
cg00035864           0.236234           0.287561           0.318016
cg00061679           0.427194           0.395514           0.456510
cg00063477           0.929039           0.927137           0.940222
cg00121626           0.481058           0.357316           0.328793
cg00223952           0.044029           0.040062           0.038420

            9247377085_R04C02  9247377093_R05C01  9247377093_R02C02
IlmnID
cg00035864           0.308176           0.239339           0.161795
cg00061679           0.525169           0.523010           0.549533
cg00063477           0.932739           0.930215           0.931468
cg00121626           0.330045           0.403873           0.313132
cg00223952           0.022201           0.027155           0.022284
```

[32]:
```
sketchy_probes_list = methylcheck.list_problem_probes('450k', ['Chen2013',
↪'Polymorphism'])
df2 = methylcheck.exclude_probes(df, sketchy_probes_list)
methylcheck.mean_beta_compare(df,df2)
```

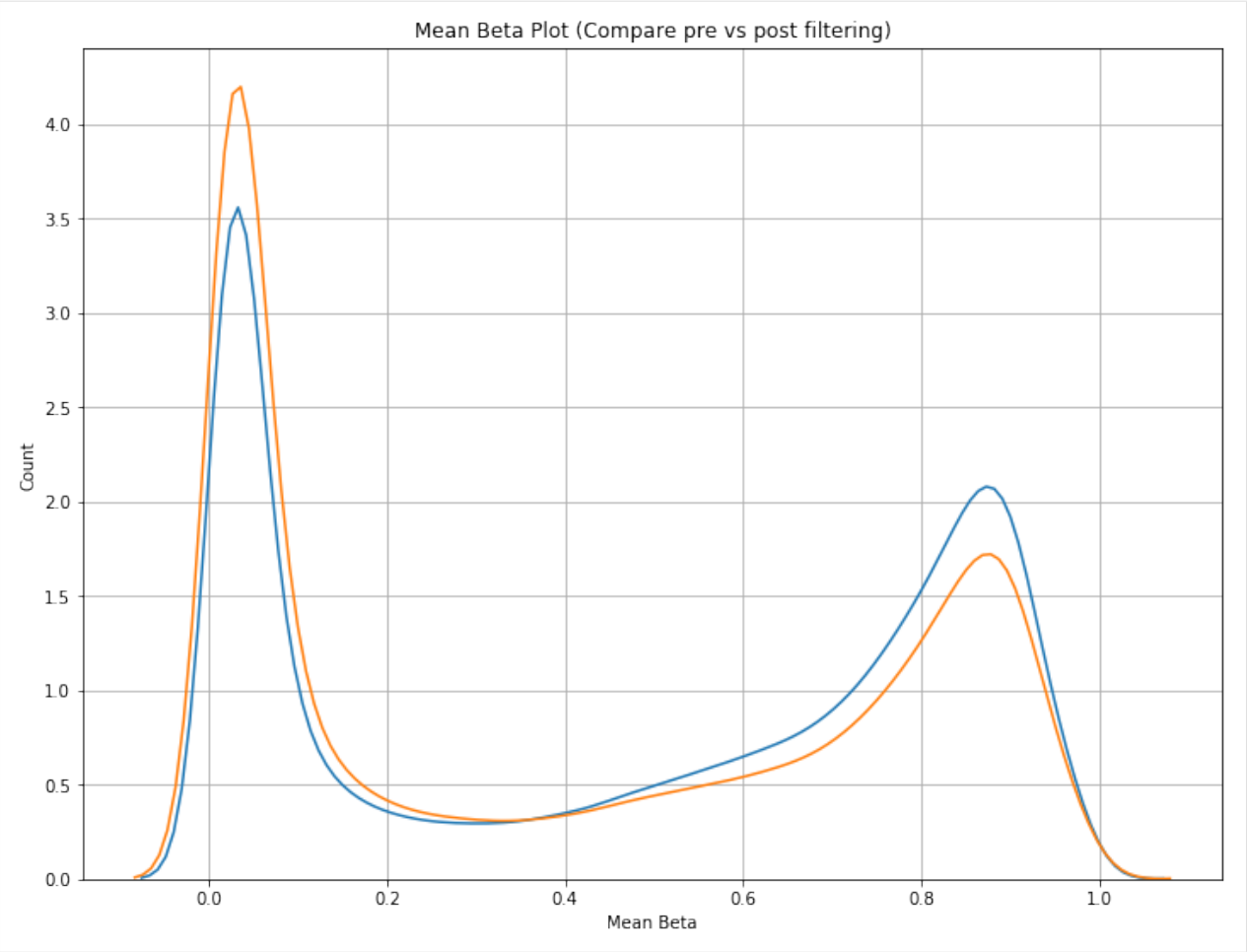
```
Of 485512 probes, 290858 matched, yielding 194654 probes after filtering.
```



4.3.1 Be careful – you can apply the a probe list for EPIC to a 450k dataset, and it will work, but won't be good filtering.

[34]:
```
sketchy_probes_list = methylcheck.list_problem_probes('EPIC', ['McCartney2016'])
df2 = methylcheck.exclude_probes(df, sketchy_probes_list)
methylcheck.mean_beta_compare(df,df2)
```

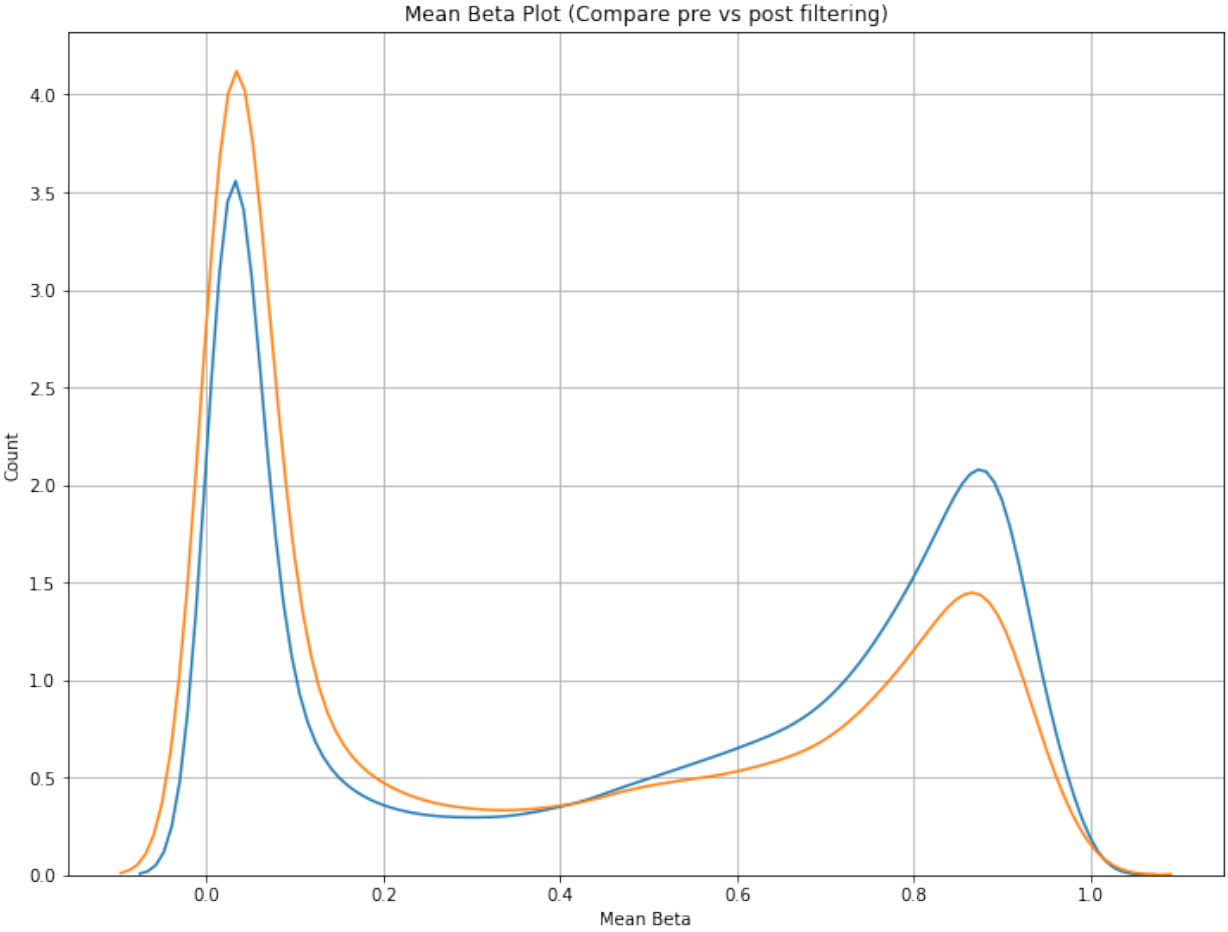
```
Of 485512 probes, 151418 matched, yielding 334094 probes after filtering.
```



[35]:
```
## Maximum filtering happens by default. (passing no criteria)
```

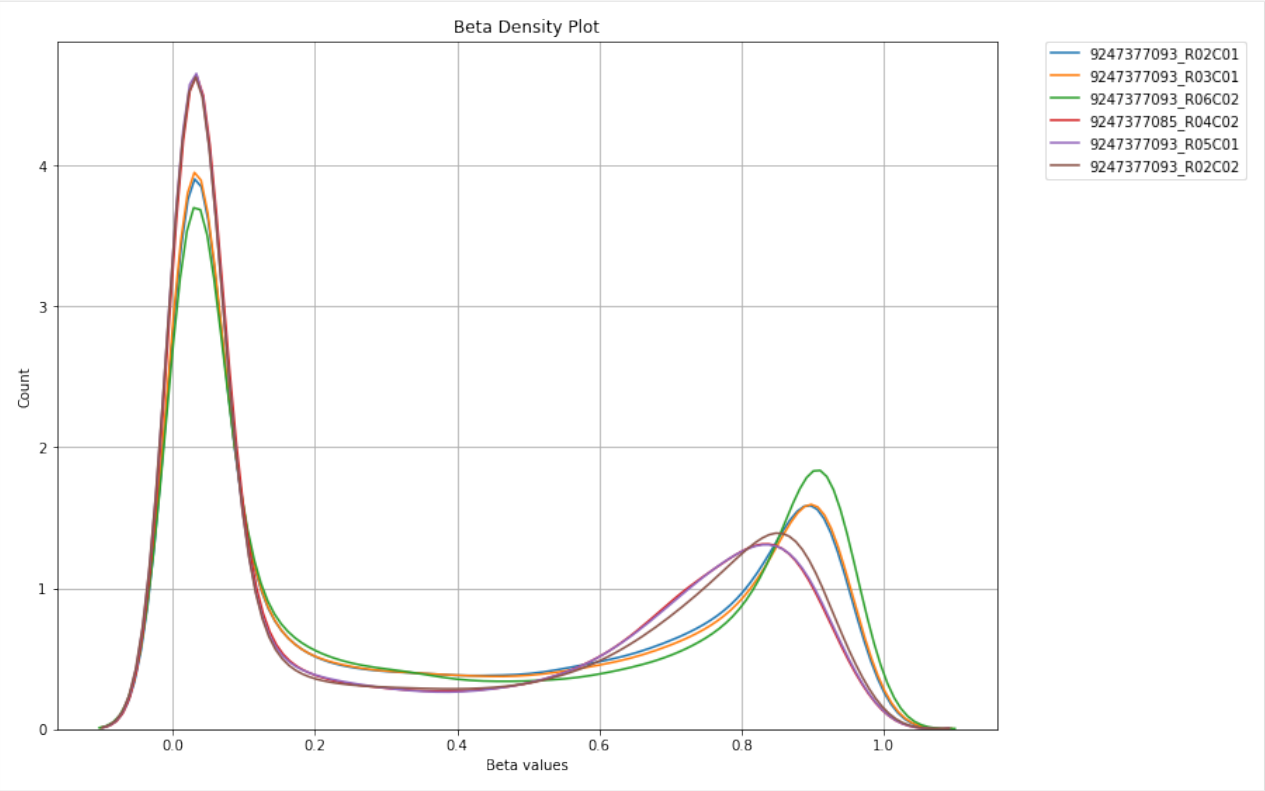
[36]:
```
sketchy_probes_list = methylcheck.list_problem_probes('450k')
df3 = methylcheck.exclude_probes(df, sketchy_probes_list)
methylcheck.mean_beta_compare(df,df3)
```

```
Of 485512 probes, 341057 matched, yielding 144455 probes after filtering.
```



[37]:
```
#underlying samples
methylcheck.beta_density_plot(df3)
```

```
6
```



There are other filtering techniques, such as MDS and cumulative_sum in other example notebooks.

[ ]:

4.4 Quality Control Example

4.4.1 using methylprep and methylcheck and GSE69852 from the GEO database

Import modules

[1]:
```
# add directory of function to path
import sys
sys.path.append("/Users/mmaxmeister/legx/methylcheck/")
sys.path.append("/Users/mmaxmeister/legx/methylprep/") # even though methylprep is␣
↪pip installed, this is still necessary on my machine (MM)
import methylprep
%matplotlib inline
```

[2]:
```
# not a module yet.
import os
print(os.getcwd())
os.chdir('/Users/mmaxmeister/legx/methylcheck')
import methylcheck
dir(methylcheck)
```

```
/Users/mmaxmeister/legx/methylcheck/docs
```

[2]:
```
['DNA_mAge_Hannum',
 'NullHandler',
 '__all__',
 '__builtins__',
 '__cached__',
 '__doc__',
 '__file__',
 '__loader__',
 '__name__',
 '__package__',
 '__path__',
 '__spec__',
 'beta_density_plot',
 'beta_mds_plot',
 'cli',
 'cumulative_sum_beta_distribution',
 'detect_array',
 'exclude_probes',
 'exclude_sex_control_probes',
 'filters',
 'getLogger',
 'list_problem_probes',
 'mean_beta_compare',
 'mean_beta_plot',
 'postprocessQC']
```

4.4.2 Importing methylprep data

[5]:
```
# print(help(methylprep.run_pipeline))
# data_dir = "/Users/mmaxmeister/legx/methylprep/docs/example_data/GSE69852"

# SAMPLE DATA
# obtain and gunzip raw data from https://www.ncbi.nlm.nih.gov/geo/download/?
↪acc=GSE100850&format=file first.
# ABOUT GSE100850: Epigenetic alterations detected in the genome of very young breast␣
↪cancer patients:
#    identification of new biomarkers

### using methylprep, you would generate a sample output file called "beta_values.pkl
↪" this way:
#data_dir = "/Users/mmaxmeister/GSE100850_RAW"
#df = methylprep.run_pipeline(data_dir, betas=True) # makes df into a betas sheet.␣
↪(the consolidate function within it)

#In this example, we've already done that. So just load the data from disk, like this:
import pandas as pd
df = pd.read_pickle("docs/example_data/beta_values.pkl")
```

4.4.3 Filtering functions

[6]:
```
# command line autodetects array type. Use this to get it / confirm it.
print(methylcheck.detect_array(df))
df.shape
```

```
EPIC
```

[6]:
```
(865859, 39)
```

[7]:
```
df = methylcheck.exclude_sex_control_probes(df, 'EPIC', verbose=True)
df.shape
```

```
Array EPIC: Removed 19627 sex linked probes and 695 internal control probes from 39␣
↪samples. 846232 probes remaining.
Discrepancy between number of probes to exclude (20322) and number actually removed␣
↪(19627): 695
It appears that your sample had no control probes, or that the control probe names␣
↪didn't match the manifest (EPIC).
```

[7]:
```
(846232, 39)
```

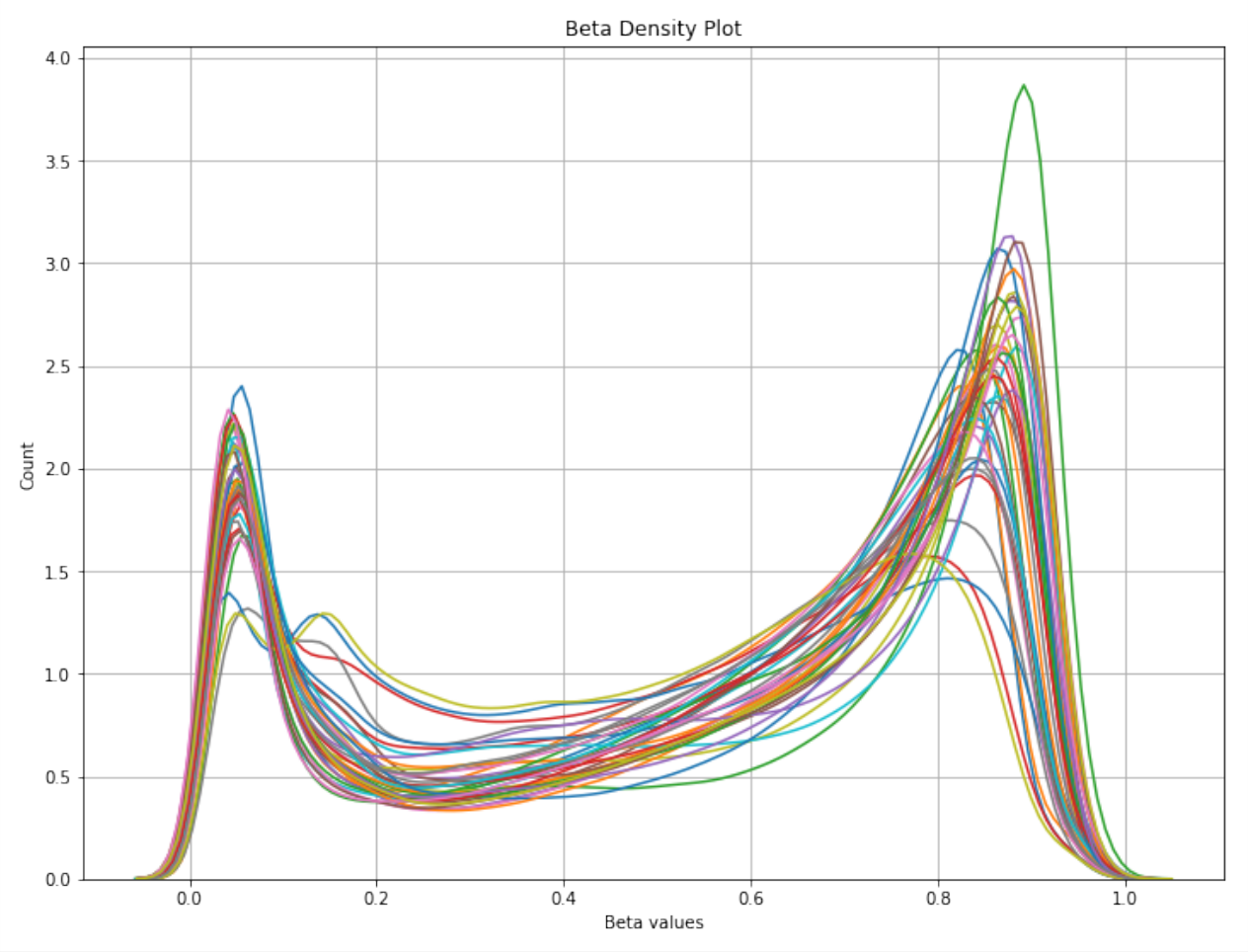
[8]:
```
#print(help(methylcheck.list_problem_probes))
# if you pass in '450k' or 'EPIC' you'll get different types of filtering.
sketchy_probes = methylcheck.list_problem_probes('EPIC')
filtered = methylcheck.filters.exclude_probes(df, sketchy_probes)
```

```
Of 846232 probes, 381361 matched, yielding 464871 probes after filtering.
```

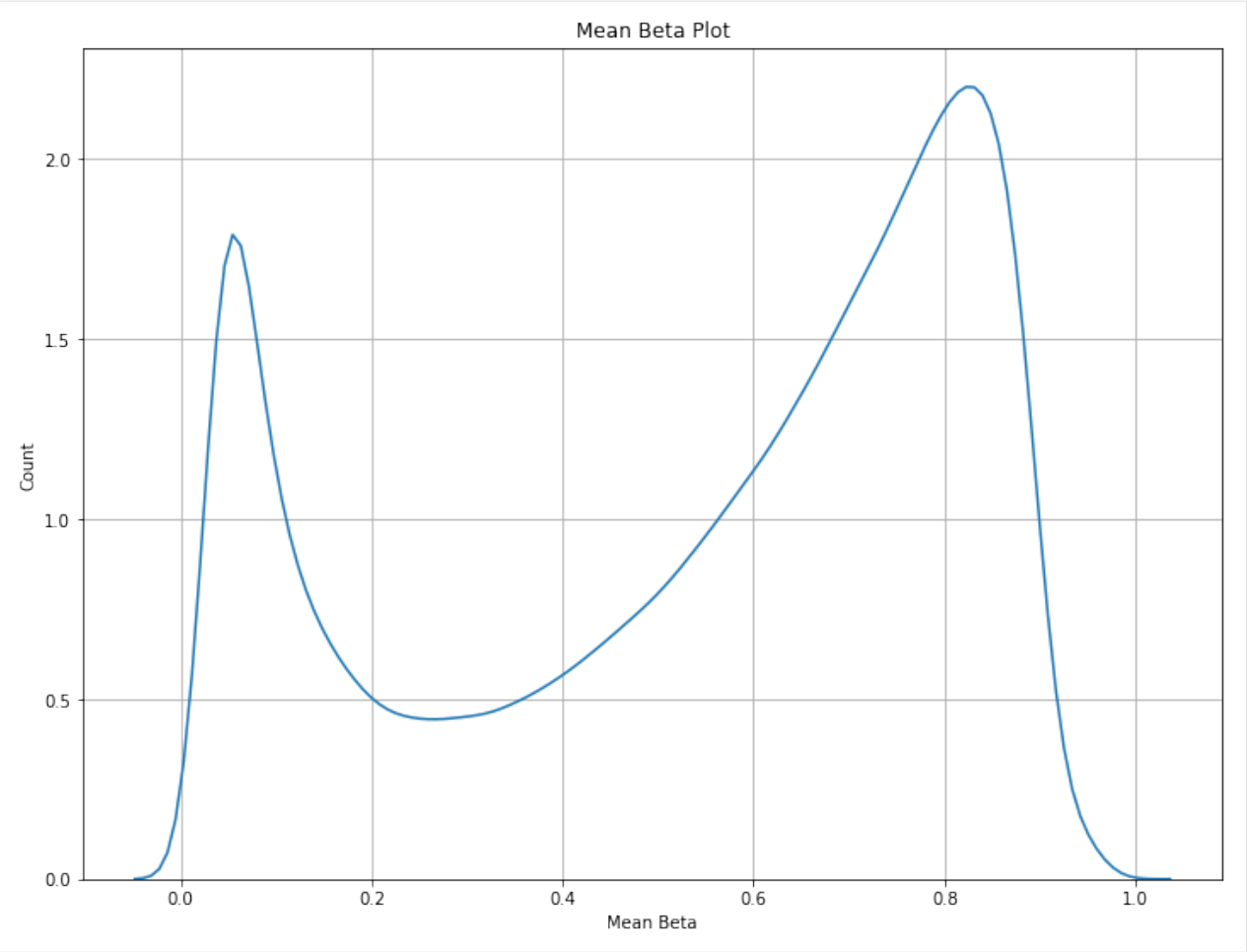
4.4.4 QC plots

[9]:
```
# Shows probe beta values for each sample as a separate trace.
methylcheck.beta_density_plot(df)
```

```
39
```



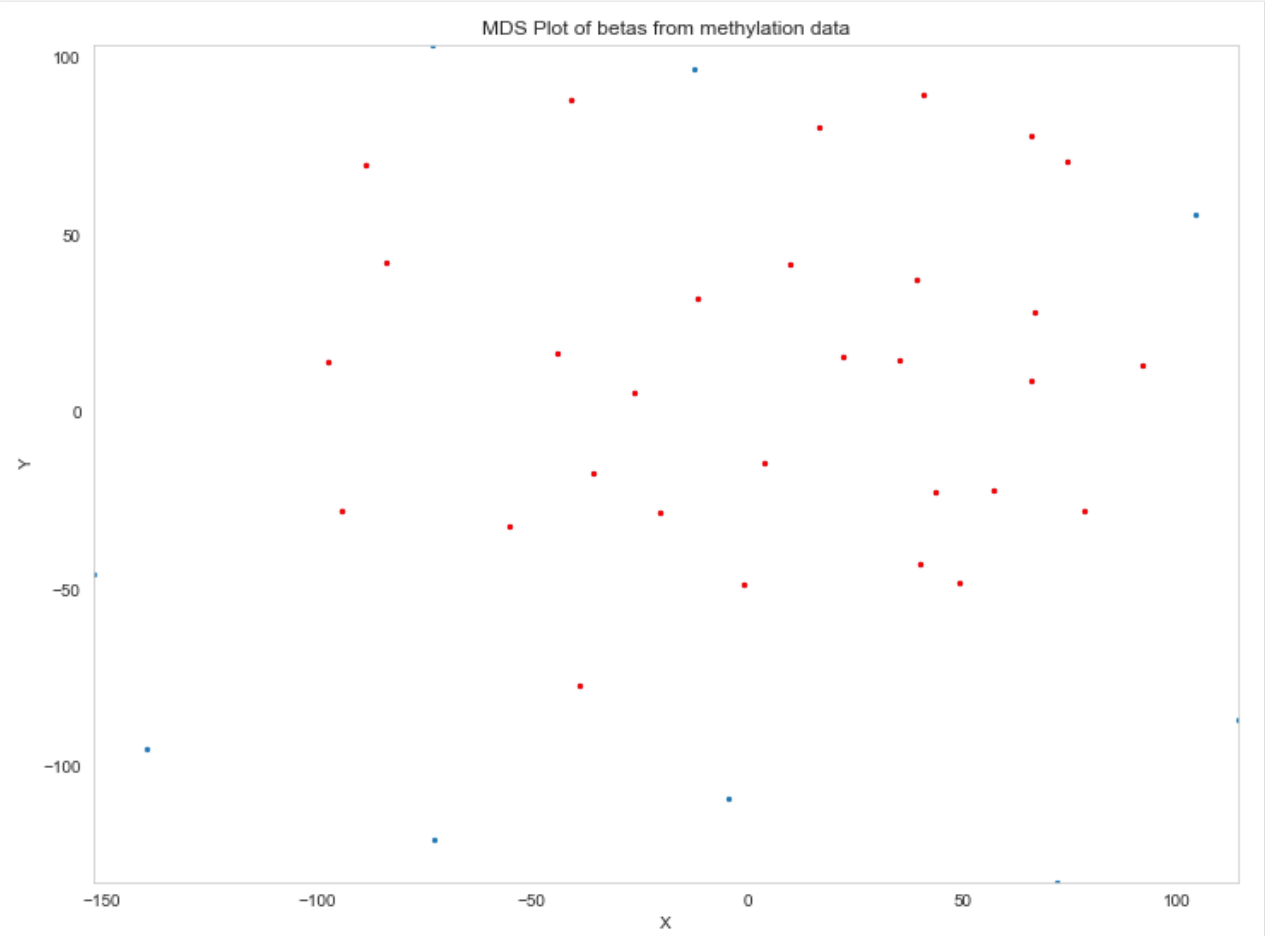
[10]:
```
# This shows all samples as a single trace.
methylcheck.mean_beta_plot(df)
```



4.4.5 Visual filtering functions

[10]:
```
# MDS filtering -- you can exclude outlier samples based on falling outside
# some standard deviation cutoff from the center of a 2D plot.
# This is an interactive function.
mds_filtered, mds_excluded = methylcheck.beta_mds_plot(df)
```

```
Your data needed to be transposed (df = df.transpose()).
(39, 846232)
Making sure that probes are in columns (the second number should be larger than the␣
↪first).
Starting MDS fit_transform. this may take a while.
You can now remove outliers based on their transformed beta values
 falling outside a range, defined by the sample standard deviation.
Your acceptable value range: x=(-100.0 to 100.0), y=(-93.0 to 93.0).
```



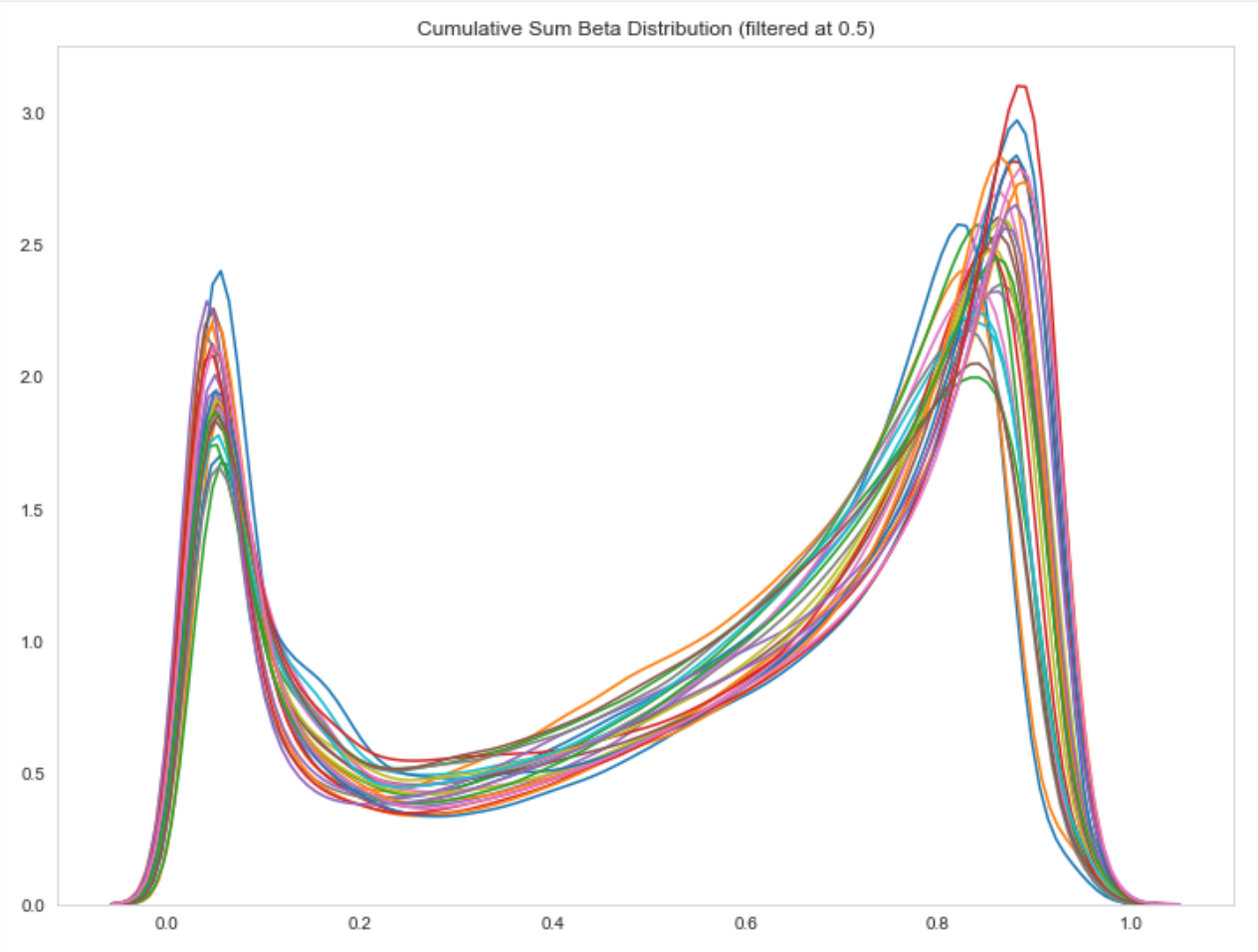

```
Original samples (39, 2) vs filtered (30, 2)
Your scale factor was: 1.5
Enter new scale factor, <enter> to accept and save:
```

[18]:
```
# adjust the cutoff value to exclude outliers. Default is 0.7.
df_outliers_removed = methylcheck.cumulative_sum_beta_distribution(mds_filtered,␣
↪cutoff=0.5)
```

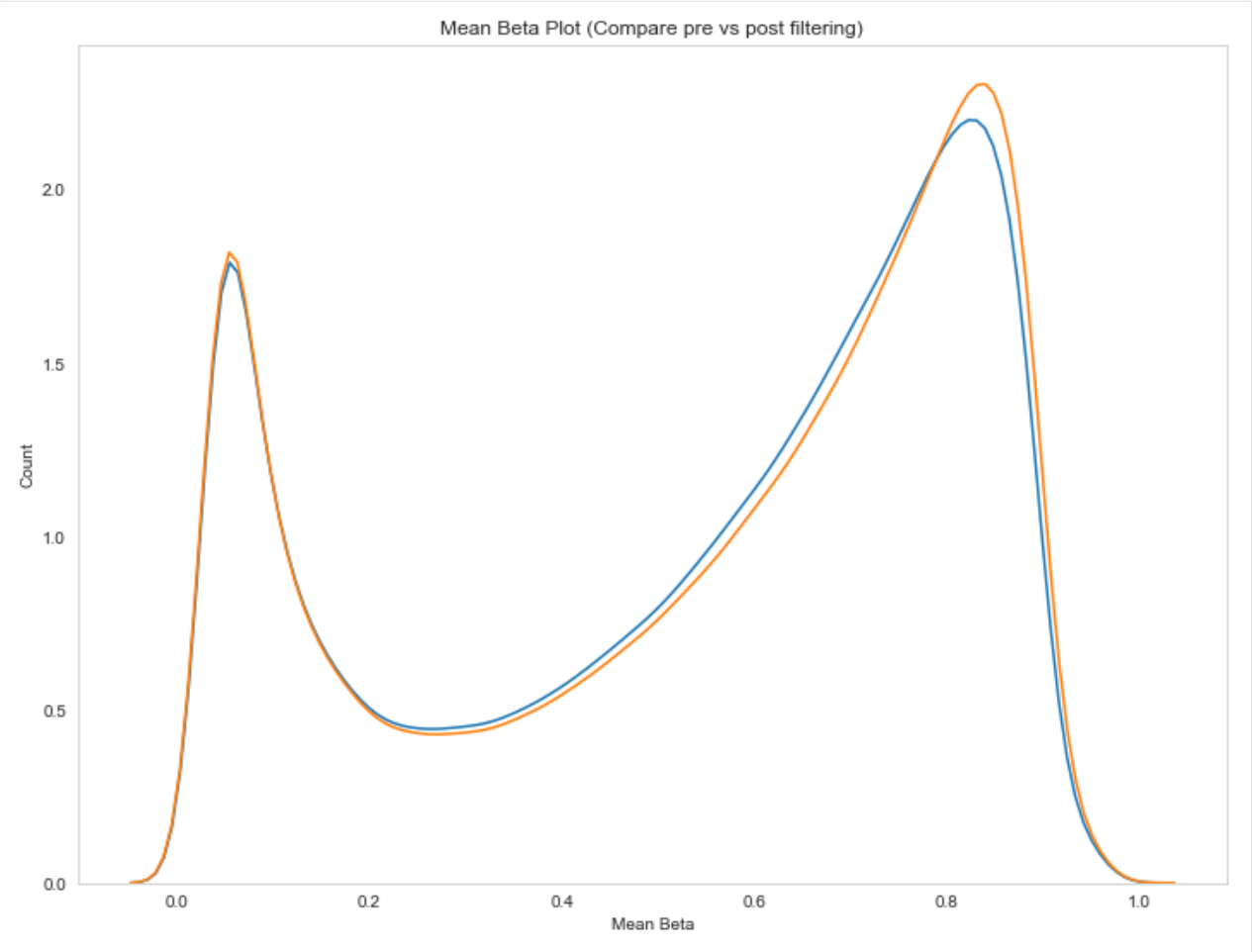
```
1it [00:00,  6.06it/s]
```

```
Calculating area under curve for each sample.
```

```
30it [00:03,  7.16it/s]
```

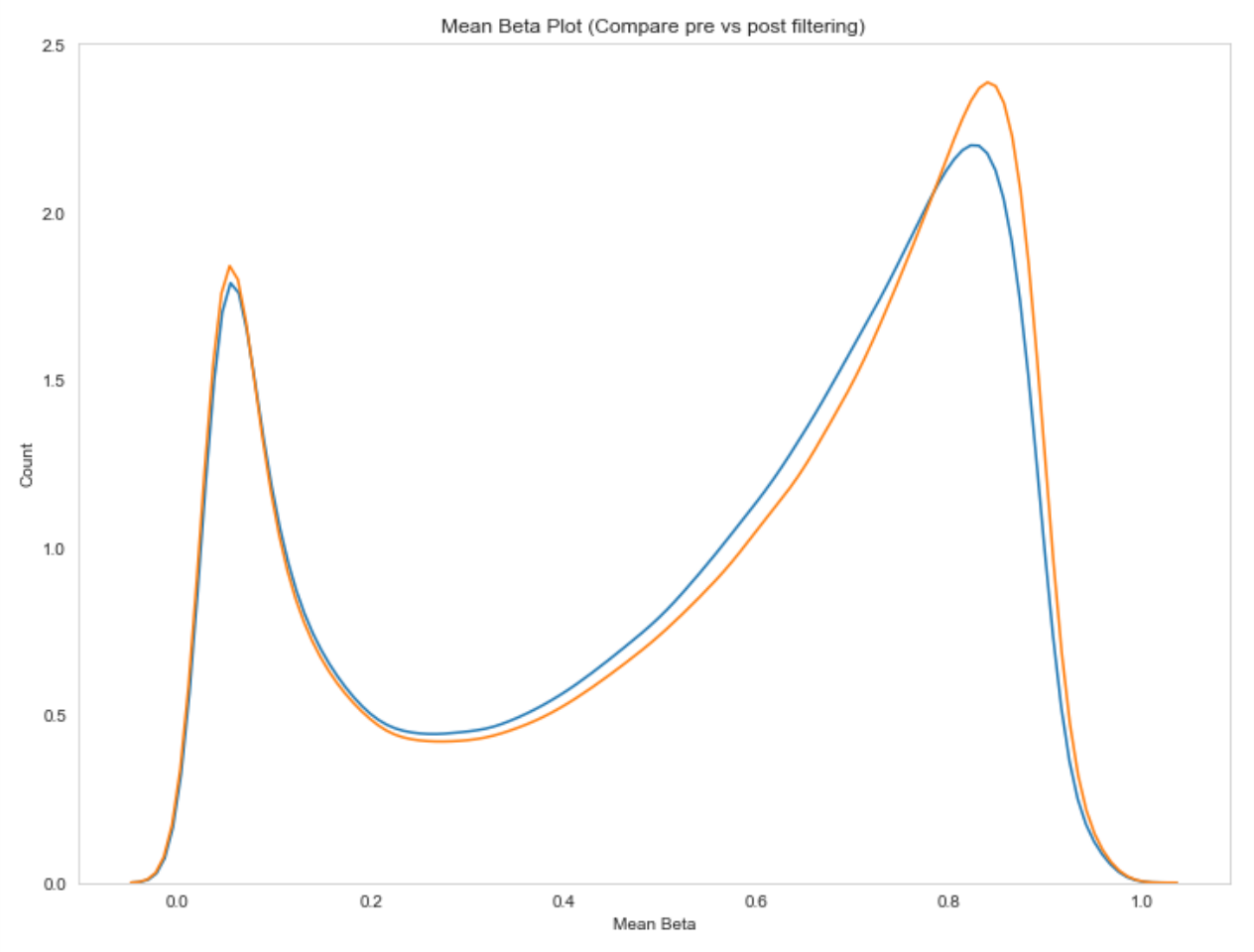


[19]:
```
# effect of MDS filtering
methylcheck.mean_beta_compare(df, mds_filtered)
```



[20]:
```
# effect of cumulative sum filtering
methylcheck.mean_beta_compare(df, df_outliers_removed, verbose=True)
```

```
Your data needed to be transposed (df = df.transpose()).
```



[ ]:

4.5 Another example of sample QC

[1]:
```
# This illustrates the simplest, smallest batch (n=2) of methylation array sample␣
↪results using methylprep and methylcheck.
#python 3.7
import pandas as pd
import numpy as np
import seaborn as sb
import matplotlib.pyplot as plt
%matplotlib inline
```

[2]:
```
%load_ext autoreload
%autoreload 2
import methylcheck
from pprint import pprint as pp
```

[4]:
```
# load sample data
from pathlib import Path
```

(continues on next page)

(continued from previous page)

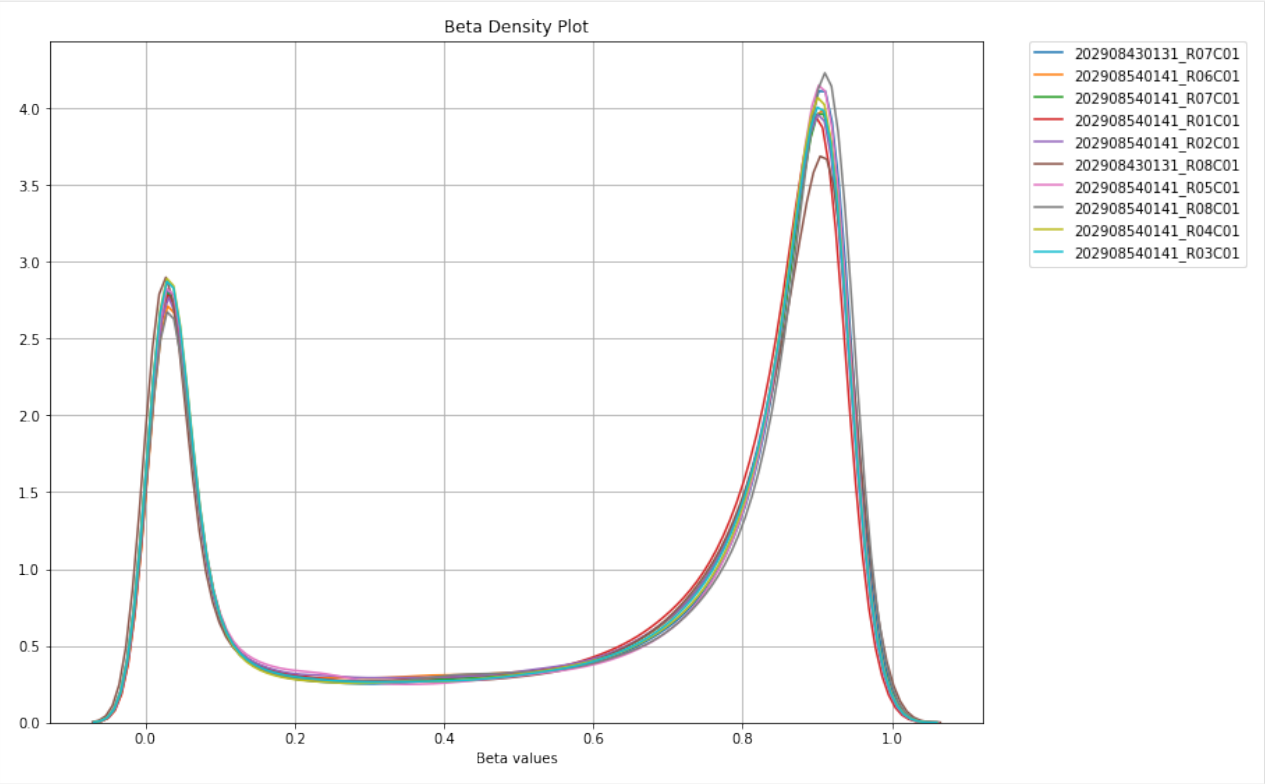
```
data_dir = '~/'
betas = pd.read_pickle(Path(data_dir, 'beta_values.pkl'))
```

[5]:
```
betas.head()
```

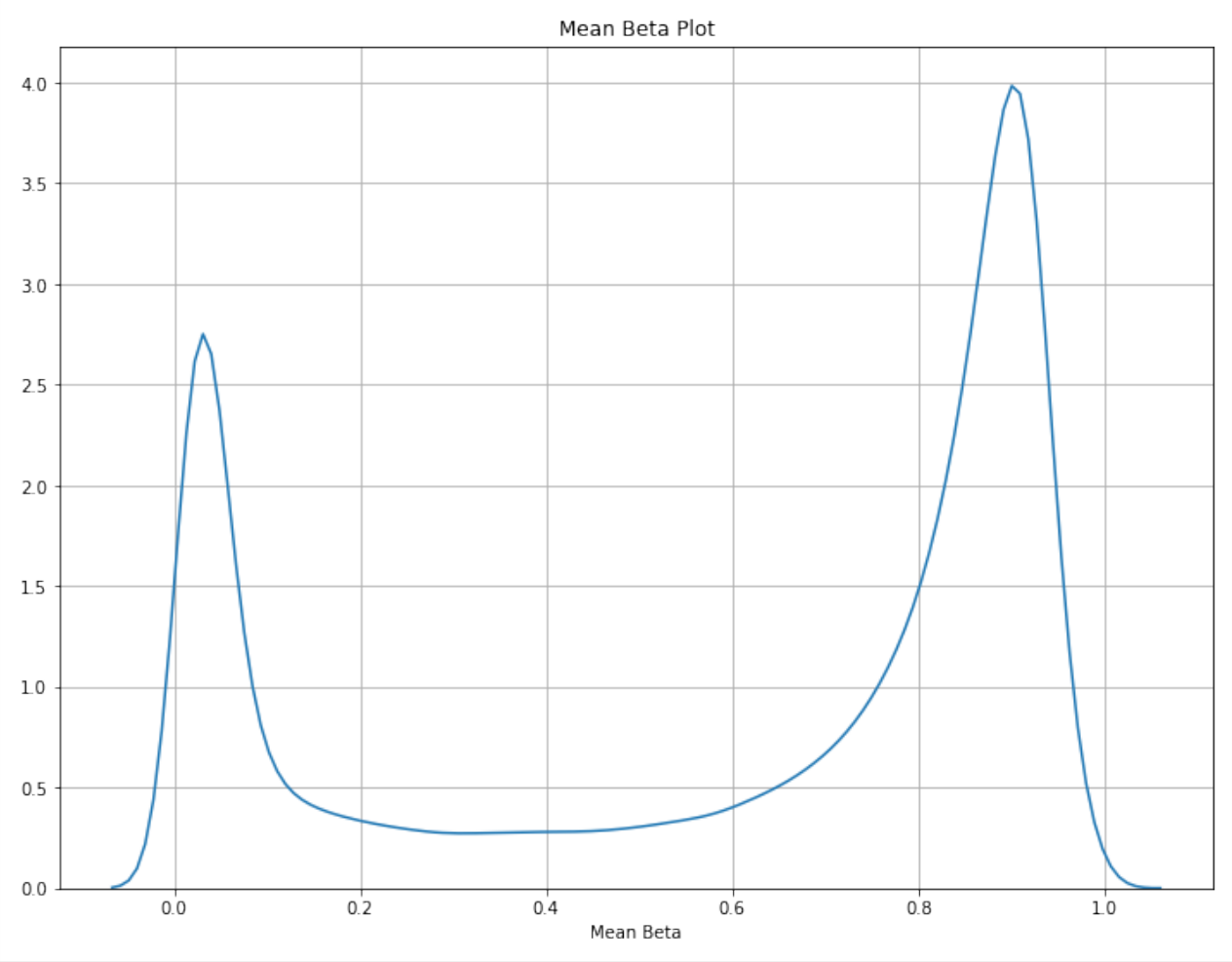
[5]:
```
            202908430131_R07C01  202908540141_R06C01  202908540141_R07C01  \
IlmnID
cg07881041             0.904343             0.900771             0.900851
cg23229610             0.923033             0.914954             0.914858
cg03513874             0.931137             0.919958             0.928000
cg05451842             0.031496             0.038683             0.031413
cg14797042             0.940138             0.934215             0.937744

            202908540141_R01C01  202908540141_R02C01  202908430131_R08C01  \
IlmnID
cg07881041             0.882696             0.909091             0.876905
cg23229610             0.902453             0.920394             0.922635
cg03513874             0.921894             0.927230             0.909573
cg05451842             0.030096             0.032964             0.020800
cg14797042             0.935057             0.939669             0.934442

            202908540141_R05C01  202908540141_R08C01  202908540141_R04C01  \
IlmnID
cg07881041             0.906337             0.919900             0.895526
cg23229610             0.925886             0.925216             0.925416
cg03513874             0.913583             0.936886             0.931464
cg05451842             0.027175             0.032345             0.029695
cg14797042             0.940236             0.943643             0.936240

            202908540141_R03C01
IlmnID
cg07881041             0.896339
cg23229610             0.920663
cg03513874             0.916364
cg05451842             0.031083
cg14797042             0.937548
```

4.5.1 beta_density_plot

[6]:
```
#plot each sample separately
methylcheck.beta_density_plot(betas)
```



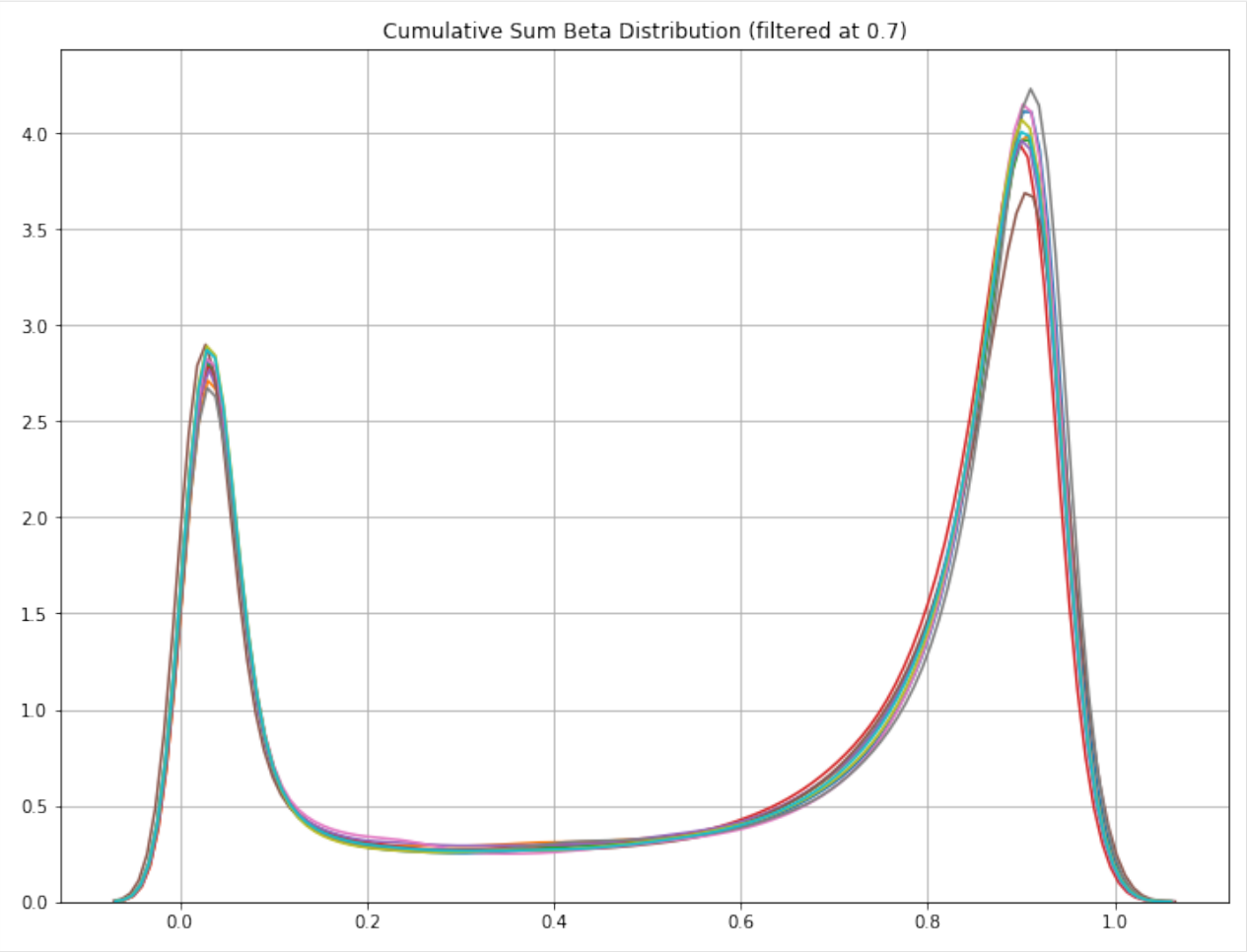
[7]:
```
# this is a mushed average of all samples into one line.
methylcheck.mean_beta_plot(betas)
```



[9]:
```
filtered_df = methylcheck.cumulative_sum_beta_distribution(betas, cutoff=0.7)
# use this to remove outliers, based on the cutoff value.
# this looks identical the beta_density_plot because no samples were removed.
```

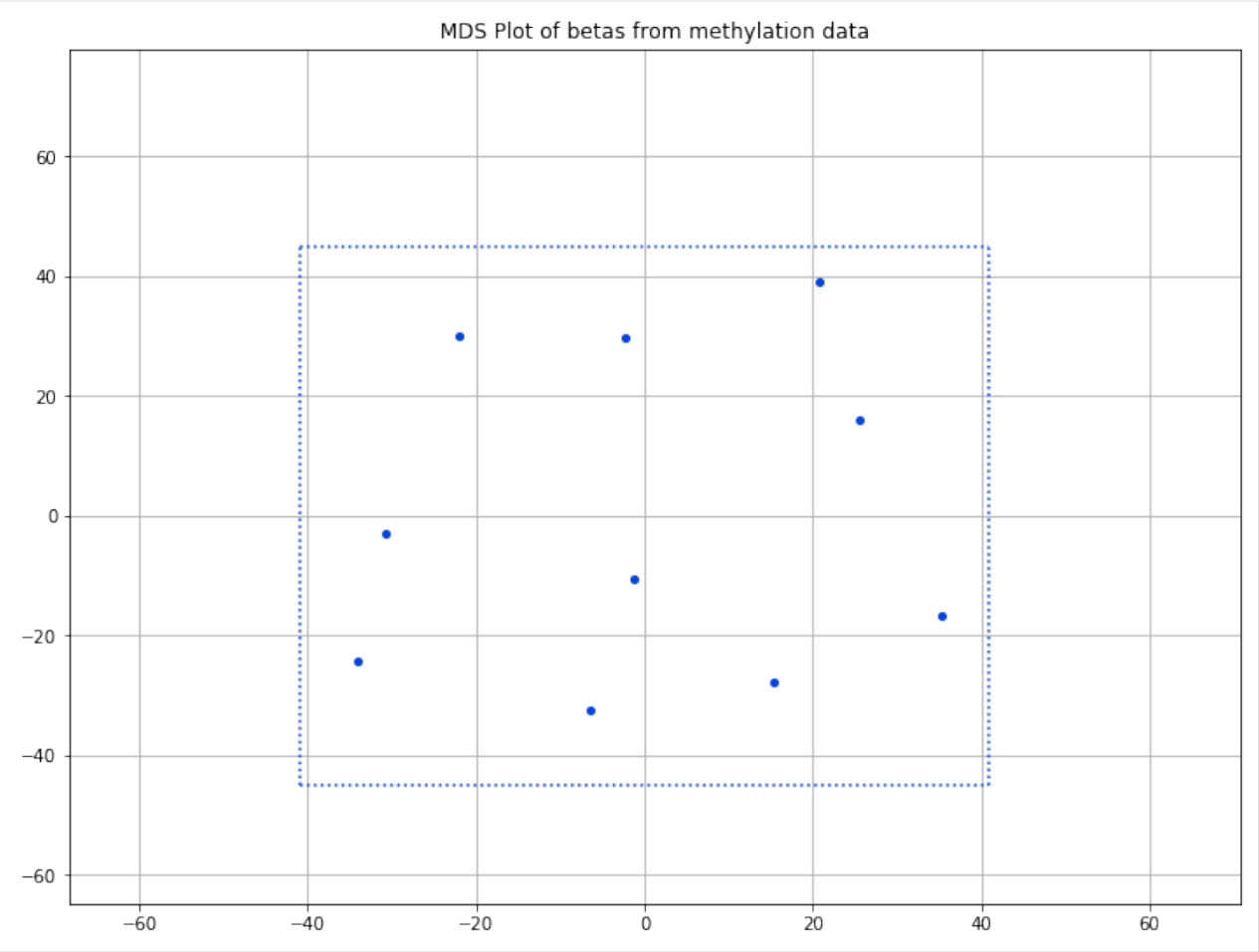
```
Calculating area under curve for each sample.
```

```
10it [00:02,  5.04it/s]
```



[13]:
```
methylcheck.beta_mds_plot(betas, filter_stdev=1.8)
```

```
Your data needed to be transposed (df = df.transpose()).
(10, 865859)
Making sure that probes are in columns (the second number should be larger than the␣
↪first).
Starting MDS fit_transform. this may take a while.
You can now remove outliers based on their transformed beta values
 falling outside a range, defined by the sample standard deviation.
Your acceptable value range: x=(-41.0 to 41.0), y=(-45.0 to 45.0).
axes None [<matplotlib.axes._subplots.AxesSubplot object at 0x7fa4c949df98>] assigned␣
↪to ax.
```




```
Original samples (0, 2) vs filtered (10, 2)
Your scale factor was: 1.8
```

```
Enter new scale factor, <enter> to accept and save:
```

[13]:
```
IlmnID               cg07881041  cg23229610  cg03513874  cg05451842  \
202908430131_R07C01    0.904343    0.923033    0.931137    0.031496
202908540141_R06C01    0.900771    0.914954    0.919958    0.038683
202908540141_R07C01    0.900851    0.914858    0.928000    0.031413
202908540141_R01C01    0.882696    0.902453    0.921894    0.030096
202908540141_R02C01    0.909091    0.920394    0.927230    0.032964
202908430131_R08C01    0.876905    0.922635    0.909573    0.020800
202908540141_R05C01    0.906337    0.925886    0.913583    0.027175
202908540141_R08C01    0.919900    0.925216    0.936886    0.032345
202908540141_R04C01    0.895526    0.925416    0.931464    0.029695
202908540141_R03C01    0.896339    0.920663    0.916364    0.031083

IlmnID               cg14797042  cg09838562  cg25458538  cg09261072  \
202908430131_R07C01    0.940138    0.032304    0.929787    0.598709
202908540141_R06C01    0.934215    0.034625    0.934069    0.618233
202908540141_R07C01    0.937744    0.030491    0.929340    0.573967
202908540141_R01C01    0.935057    0.033836    0.923905    0.616446
202908540141_R02C01    0.939669    0.029871    0.922341    0.594974
202908430131_R08C01    0.934442    0.042985    0.928857    0.578807
202908540141_R05C01    0.940236    0.032446    0.927417    0.595427
202908540141_R08C01    0.943643    0.036972    0.933692    0.607197
```

(continues on next page)

(continued from previous page)

```
202908540141_R04C01    0.936240    0.035692    0.921008    0.570685
202908540141_R03C01    0.937548    0.037391    0.934201    0.575947

IlmnID               cg02404579  cg04118974  ...  cg22005990  cg05384275  \
202908430131_R07C01    0.845217    0.650180  ...    0.151468    0.013035
202908540141_R06C01    0.854144    0.629472  ...    0.263880    0.014193
202908540141_R07C01    0.823189    0.577657  ...    0.225832    0.015058
202908540141_R01C01    0.842846    0.606424  ...    0.234230    0.016249
202908540141_R02C01    0.826403    0.571482  ...    0.173193    0.015260
202908430131_R08C01    0.872678    0.597685  ...    0.199412    0.009801
202908540141_R05C01    0.827010    0.507028  ...    0.108926    0.015647
202908540141_R08C01    0.857196    0.644634  ...    0.198390    0.013094
202908540141_R04C01    0.836864    0.537062  ...    0.089540    0.014230
202908540141_R03C01    0.803317    0.546901  ...    0.138550    0.013856

IlmnID               cg21496658  cg27017993  cg19551589  cg10218605  \
202908430131_R07C01    0.013853    0.916667    0.027761    0.387055
202908540141_R06C01    0.021349    0.903981    0.031160    0.463987
202908540141_R07C01    0.014798    0.919489    0.076705    0.387620
202908540141_R01C01    0.025378    0.900429    0.028450    0.511480
202908540141_R02C01    0.018583    0.900903    0.029741    0.415422
202908430131_R08C01    0.015225    0.907851    0.028729    0.550254
202908540141_R05C01    0.017776    0.907857    0.028055    0.595931
202908540141_R08C01    0.016988    0.915133    0.026774    0.466936
202908540141_R04C01    0.016273    0.914756    0.032220    0.650614
202908540141_R03C01    0.015158    0.912915    0.033248    0.448766

IlmnID               cg06899844  cg22494081  cg22623303  cg21064505
202908430131_R07C01    0.035152    0.970914    0.972975    0.955782
202908540141_R06C01    0.040007    0.966107    0.965929    0.949739
202908540141_R07C01    0.035784    0.969959    0.966356    0.956764
202908540141_R01C01    0.041063    0.963544    0.963379    0.949240
202908540141_R02C01    0.038572    0.968489    0.965089    0.953391
202908430131_R08C01    0.034694    0.974369    0.969074    0.956301
202908540141_R05C01    0.037338    0.966914    0.967480    0.948400
202908540141_R08C01    0.033210    0.973266    0.968754    0.960836
202908540141_R04C01    0.041320    0.967863    0.967791    0.953156
202908540141_R03C01    0.040269    0.965765    0.965241    0.957500

[10 rows x 865859 columns]
```

[ ]:

[29]:
```
import pandas as pd
```

4.6 Orienting dataframes to match

[ ]:
```
"""the trick is to ensure the index matches, or that both contain one column for␣
↪merging.
"""
```

[30]:
```
meta = pd.read_pickle('sample_Sheet_meta_data.pkl')
meta
```

[30]:
```
  Sample_Type    Sentrix_ID Sentrix_Position Sample_Group Sample_Name  \
0     Unknown  202908430131           R07C01         None      CTRL01
1     Unknown  202908540141           R06C01         None      CTRL04
2     Unknown  202908540141           R07C01         None      CTRL05
3     Unknown  202908540141           R01C01         None          L4
4     Unknown  202908540141           R02C01         None          L2
5     Unknown  202908430131           R08C01         None      CTRL02
6     Unknown  202908540141           R05C01         None      CTRL03
7     Unknown  202908540141           R08C01         None      CTRL06
8     Unknown  202908540141           R04C01         None          L3
9     Unknown  202908540141           R03C01         None          L1

  Sample_Plate Sample_Type Sub_Type Sample_Well Pool_ID      GSM_ID Control  \
0         None     Unknown     None         None    None  GSM3927205   False
1         None     Unknown     None         None    None  GSM3927208   False
2         None     Unknown     None         None    None  GSM3927209   False
3         None     Unknown     None         None    None  GSM3927214   False
4         None     Unknown     None         None    None  GSM3927212   False
5         None     Unknown     None         None    None  GSM3927206   False
6         None     Unknown     None         None    None  GSM3927207   False
7         None     Unknown     None         None    None  GSM3927210   False
8         None     Unknown     None         None    None  GSM3927213   False
9         None     Unknown     None         None    None  GSM3927211   False

             Sample_ID
0  202908430131_R07C01
1  202908540141_R06C01
2  202908540141_R07C01
3  202908540141_R01C01
4  202908540141_R02C01
5  202908430131_R08C01
6  202908540141_R05C01
7  202908540141_R08C01
8  202908540141_R04C01
9  202908540141_R03C01
```

[31]:
```
m_values = pd.read_pickle('m_values.pkl')
m_values.head()
```

[31]:
```
            202908430131_R07C01  202908540141_R06C01  202908540141_R07C01  \
IlmnID
cg07881041             3.412787             3.345855             3.349728
cg23229610             4.038538             3.865307             3.852169
cg03513874             4.116045             3.840853             4.047271
cg05451842            -4.910128            -4.602309            -4.914334
cg14797042             4.464024             4.292200             4.383563

            202908540141_R01C01  202908540141_R02C01  202908430131_R08C01  \
IlmnID
cg07881041             3.082507             3.542418             2.960507
cg23229610             3.671637             4.042739             4.180840
cg03513874             3.949184             4.078279             3.633142
cg05451842            -4.967814            -4.836660            -5.514546
cg14797042             4.382857             4.562915             4.372119

            202908540141_R05C01  202908540141_R08C01  202908540141_R04C01  \
IlmnID
```

(continues on next page)

(continued from previous page)

```
cg07881041             3.465868             3.729893             3.263243
cg23229610             4.124988             4.124023             4.165693
cg03513874             3.701213             4.302513             4.155497
cg05451842            -5.124825            -4.866898            -4.993714
cg14797042             4.479269             4.617060             4.424321

            202908540141_R03C01
IlmnID
cg07881041             3.290433
cg23229610             4.018463
cg03513874             3.791341
cg05451842            -4.925330
cg14797042             4.435270
```

[32]:
```
meta = meta.set_index('Sample_ID')
```

[36]:
```
m_values = m_values.transpose() if m_values.shape[1] < m_values.shape[0] else m_
↪values # ensure structure same
merged_df = m_values.merge(meta, left_index=True, right_index=True)
```

[37]:
```
meta
```

[37]:
```
                    Sample_Type    Sentrix_ID Sentrix_Position Sample_Group  \
Sample_ID
202908430131_R07C01     Unknown  202908430131           R07C01         None
202908540141_R06C01     Unknown  202908540141           R06C01         None
202908540141_R07C01     Unknown  202908540141           R07C01         None
202908540141_R01C01     Unknown  202908540141           R01C01         None
202908540141_R02C01     Unknown  202908540141           R02C01         None
202908430131_R08C01     Unknown  202908430131           R08C01         None
202908540141_R05C01     Unknown  202908540141           R05C01         None
202908540141_R08C01     Unknown  202908540141           R08C01         None
202908540141_R04C01     Unknown  202908540141           R04C01         None
202908540141_R03C01     Unknown  202908540141           R03C01         None

                    Sample_Name Sample_Plate Sample_Type Sub_Type Sample_Well  \
Sample_ID
202908430131_R07C01      CTRL01         None     Unknown     None        None
202908540141_R06C01      CTRL04         None     Unknown     None        None
202908540141_R07C01      CTRL05         None     Unknown     None        None
202908540141_R01C01          L4         None     Unknown     None        None
202908540141_R02C01          L2         None     Unknown     None        None
202908430131_R08C01      CTRL02         None     Unknown     None        None
202908540141_R05C01      CTRL03         None     Unknown     None        None
202908540141_R08C01      CTRL06         None     Unknown     None        None
202908540141_R04C01          L3         None     Unknown     None        None
202908540141_R03C01          L1         None     Unknown     None        None

                    Pool_ID      GSM_ID Control
Sample_ID
202908430131_R07C01    None  GSM3927205   False
202908540141_R06C01    None  GSM3927208   False
202908540141_R07C01    None  GSM3927209   False
202908540141_R01C01    None  GSM3927214   False
202908540141_R02C01    None  GSM3927212   False
202908430131_R08C01    None  GSM3927206   False
```

(continues on next page)

(continued from previous page)

```
202908540141_R05C01    None  GSM3927207   False
202908540141_R08C01    None  GSM3927210   False
202908540141_R04C01    None  GSM3927213   False
202908540141_R03C01    None  GSM3927211   False
```

[38]:
```
m_values
```

[38]:
```
IlmnID               cg07881041  cg23229610  cg03513874  cg05451842  \
202908430131_R07C01    3.412787    4.038538    4.116045   -4.910128
202908540141_R06C01    3.345855    3.865307    3.840853   -4.602309
202908540141_R07C01    3.349728    3.852169    4.047271   -4.914334
202908540141_R01C01    3.082507    3.671637    3.949184   -4.967814
202908540141_R02C01    3.542418    4.042739    4.078279   -4.836660
202908430131_R08C01    2.960507    4.180840    3.633142   -5.514546
202908540141_R05C01    3.465868    4.124988    3.701213   -5.124825
202908540141_R08C01    3.729893    4.124023    4.302513   -4.866898
202908540141_R04C01    3.263243    4.165693    4.155497   -4.993714
202908540141_R03C01    3.290433    4.018463    3.791341   -4.925330

IlmnID               cg14797042  cg09838562  cg25458538  cg09261072  \
202908430131_R07C01    4.464024   -4.850363    3.972917    0.608777
202908540141_R06C01    4.292200   -4.745925    4.091592    0.729237
202908540141_R07C01    4.383563   -4.937930    3.948131    0.458526
202908540141_R01C01    4.382857   -4.760096    3.880529    0.724771
202908540141_R02C01    4.562915   -4.963109    3.829859    0.591076
202908430131_R08C01    4.372119   -4.384490    3.946780    0.490380
202908540141_R05C01    4.479269   -4.845135    3.916670    0.590274
202908540141_R08C01    4.617060   -4.646371    4.056437    0.661098
202908540141_R04C01    4.424321   -4.701726    3.771189    0.440081
202908540141_R03C01    4.435270   -4.627422    4.117044    0.474540

IlmnID               cg02404579  cg04118974  ...  cg22005990  cg05384275  \
202908430131_R07C01    2.593320    0.957829  ...   -2.463625   -6.222727
202908540141_R06C01    2.690233    0.826782  ...   -1.456612   -6.098733
202908540141_R07C01    2.325352    0.504637  ...   -1.755983   -6.012443
202908540141_R01C01    2.592081    0.699158  ...   -1.679747   -5.896445
202908540141_R02C01    2.381769    0.477761  ...   -2.228573   -5.989043
202908430131_R08C01    2.962171    0.635061  ...   -1.980125   -6.638985
202908540141_R05C01    2.374042    0.087433  ...   -3.009043   -5.955131
202908540141_R08C01    2.730637    0.926858  ...   -1.991519   -6.215806
202908540141_R04C01    2.481938    0.269274  ...   -3.321619   -6.094163
202908540141_R03C01    2.137687    0.330344  ...   -2.611546   -6.131170

IlmnID               cg21496658  cg27017993  cg19551589  cg10218605  \
202908430131_R07C01   -6.135573    3.945602   -5.102091   -0.631836
202908540141_R06C01   -5.501676    3.735050   -4.929893   -0.175451
202908540141_R07C01   -6.038302    4.025450   -3.566083   -0.630898
202908540141_R01C01   -5.243735    3.737237   -5.057497    0.113837
202908540141_R02C01   -5.702124    3.718148   -4.994833   -0.454764
202908430131_R08C01   -5.998120    3.912966   -5.049951    0.338431
202908540141_R05C01   -5.769887    3.774773   -5.085601    0.606516
202908540141_R08C01   -5.837299    3.949614   -5.154335   -0.154815
202908540141_R04C01   -5.897993    3.959106   -4.880647    0.943185
202908540141_R03C01   -5.999020    3.935216   -4.829910   -0.259069

IlmnID               cg06899844  cg22494081  cg22623303  cg21064505
202908430131_R07C01   -4.743207    5.585948    5.890584    5.112240
```

(continues on next page)

(continued from previous page)

```
202908540141_R06C01   -4.550831    5.341666    5.371025    4.830027
202908540141_R07C01   -4.719253    5.517068    5.454447    5.133564
202908540141_R01C01   -4.504871    5.290402    5.445842    4.979013
202908540141_R02C01   -4.600786    5.529454    5.507032    5.079351
202908430131_R08C01   -4.757568    5.891426    5.677103    5.228642
202908540141_R05C01   -4.654901    5.387020    5.523357    4.829708
202908540141_R08C01   -4.827380    5.775283    5.607601    5.354515
202908540141_R04C01   -4.500393    5.433155    5.592485    5.004386
202908540141_R03C01   -4.535684    5.345188    5.456787    5.241151

[10 rows x 865859 columns]
```

[39]:
```
merged_df
```

[39]:
```
                     cg07881041  cg23229610  cg03513874  cg05451842  \
202908430131_R07C01    3.412787    4.038538    4.116045   -4.910128
202908540141_R06C01    3.345855    3.865307    3.840853   -4.602309
202908540141_R07C01    3.349728    3.852169    4.047271   -4.914334
202908540141_R01C01    3.082507    3.671637    3.949184   -4.967814
202908540141_R02C01    3.542418    4.042739    4.078279   -4.836660
202908430131_R08C01    2.960507    4.180840    3.633142   -5.514546
202908540141_R05C01    3.465868    4.124988    3.701213   -5.124825
202908540141_R08C01    3.729893    4.124023    4.302513   -4.866898
202908540141_R04C01    3.263243    4.165693    4.155497   -4.993714
202908540141_R03C01    3.290433    4.018463    3.791341   -4.925330

                     cg14797042  cg09838562  cg25458538  cg09261072  \
202908430131_R07C01    4.464024   -4.850363    3.972917    0.608777
202908540141_R06C01    4.292200   -4.745925    4.091592    0.729237
202908540141_R07C01    4.383563   -4.937930    3.948131    0.458526
202908540141_R01C01    4.382857   -4.760096    3.880529    0.724771
202908540141_R02C01    4.562915   -4.963109    3.829859    0.591076
202908430131_R08C01    4.372119   -4.384490    3.946780    0.490380
202908540141_R05C01    4.479269   -4.845135    3.916670    0.590274
202908540141_R08C01    4.617060   -4.646371    4.056437    0.661098
202908540141_R04C01    4.424321   -4.701726    3.771189    0.440081
202908540141_R03C01    4.435270   -4.627422    4.117044    0.474540

                     cg02404579  cg04118974  ...  Sentrix_Position  \
202908430131_R07C01    2.593320    0.957829  ...            R07C01
202908540141_R06C01    2.690233    0.826782  ...            R06C01
202908540141_R07C01    2.325352    0.504637  ...            R07C01
202908540141_R01C01    2.592081    0.699158  ...            R01C01
202908540141_R02C01    2.381769    0.477761  ...            R02C01
202908430131_R08C01    2.962171    0.635061  ...            R08C01
202908540141_R05C01    2.374042    0.087433  ...            R05C01
202908540141_R08C01    2.730637    0.926858  ...            R08C01
202908540141_R04C01    2.481938    0.269274  ...            R04C01
202908540141_R03C01    2.137687    0.330344  ...            R03C01

                     Sample_Group  Sample_Name  Sample_Plate  Sample_Type  \
202908430131_R07C01          None       CTRL01          None      Unknown
202908540141_R06C01          None       CTRL04          None      Unknown
202908540141_R07C01          None       CTRL05          None      Unknown
202908540141_R01C01          None           L4          None      Unknown
202908540141_R02C01          None           L2          None      Unknown
202908430131_R08C01          None       CTRL02          None      Unknown
```

(continues on next page)

(continued from previous page)

```
202908540141_R05C01          None       CTRL03          None      Unknown
202908540141_R08C01          None       CTRL06          None      Unknown
202908540141_R04C01          None           L3          None      Unknown
202908540141_R03C01          None           L1          None      Unknown

                     Sub_Type  Sample_Well  Pool_ID      GSM_ID  Control
202908430131_R07C01      None         None     None  GSM3927205    False
202908540141_R06C01      None         None     None  GSM3927208    False
202908540141_R07C01      None         None     None  GSM3927209    False
202908540141_R01C01      None         None     None  GSM3927214    False
202908540141_R02C01      None         None     None  GSM3927212    False
202908430131_R08C01      None         None     None  GSM3927206    False
202908540141_R05C01      None         None     None  GSM3927207    False
202908540141_R08C01      None         None     None  GSM3927210    False
202908540141_R04C01      None         None     None  GSM3927213    False
202908540141_R03C01      None         None     None  GSM3927211    False

[10 rows x 865871 columns]
```

4.7 methylcheck package

4.7.1 methylcheck.cli module

class methylcheck.cli.**DefaultParser**(*prog=None*, *usage=None*, *description=None*, *epilog=None*, *parents=[]*, *formatter_class=<class 'argparse.HelpFormatter'>*, *prefix_chars='-'*, *fromfile_prefix_chars=None*, *argument_default=None*, *conflict_handler='error'*, *add_help=True*, *allow_abbrev=True*)

Bases: argparse.ArgumentParser

error(*message: string*)

Prints a usage message incorporating the message to stderr and exits.

If you override this in a subclass, it should not return – it should either exit or raise an exception.

methylcheck.cli.**cli_parser**()

methylcheck.cli.**detect_array**(*df*)

Determines array type using number of probes columns in df. Returns array string.

4.7.2 methylcheck.filters module

methylcheck.filters.**exclude_probes**(*array*, *probe_list*)

How to: use list_problem_probes to obtain a list of 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.

methylcheck.filters.**exclude_sex_control_probes**(*df*, *array*, *no_sex=True*, *no_control=True*, *verbose=False*)

Function to 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'} or {'IlluminaHumanMethylation27k','IlluminaHumanMethylation450k','IlluminaHumanMethylati

***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.

methylcheck.filters.**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.

array: string name for type of array used 'IlluminaHumanMethylationEPIC', 'IlluminaHumanMethylation450k' This shorthand names are also okay: 'EPIC','EPIC+','450k','27k'

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' 'McCarthey2016'

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' 'CrossHybridizing' 'BaseColorChange' 'RepeatSequenceElements'

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.

probe_exclusion_list: list List containing probe identifiers to be excluded

4.7.3 methylcheck.postprocessQC module

methylcheck.postprocessQC.**beta_density_plot**(*df*, *verbose=False*, *save=False*, *silent=False*)

Returns a plot of beta values for each sample in a batch of samples as a separate line. Y-axis values is the count (of what? intensity? normalized?). 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

methylcheck.postprocessQC.**beta_mds_plot**(*df*, *filter_stdev=1.5*, *verbose=True*, *save=False*, *silent=False*, *multi_params={'draw_box': True}*)

1 needs to read the manifest file for the array, or at least a list of probe names to exclude/include.
manifest_file = pd.read_csv('/Users/nrigby/GitHub/stp-prelim-analysis/working_data/CombinedManifestEPIC.manifest.Co 'CHR']] probe_names_no_sex_probes = manifest_file.loc[manifest_file['CHR'].apply(lambda x: x not in ['X', 'Y', np.nan]), 'IlmnID'].values probe_names_sex_probes = manifest_file.loc[manifest_file['CHR'].apply(lambda x: x in ['X', 'Y']), 'IlmnID'].values

df_no_sex_probes = df[probe_names_no_sex_probes] df_no_sex_probes.head()

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.

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)}

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. Silent also suppresses plots (images) from being generated. only files are returned.

returns a filtered dataframe. if *return_plot_obj* is True, it returns the plot, for making overlays in methylize.

pandas, numpy, pyplot, sklearn.manifold.MDS

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

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
- **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, defaulting to 1.5 STDEV.)

if you increase this number, fewer outlier samples will be removed.

- TODO: one dataframe of the retained samples, cutoff box is avg of datasets

~~nothing returned (currently)~~ - TODO: each dataset's results as a transformed file - default: list of samples retained or excluded - option: a list of pyplot subplot objects

methylcheck.postprocessQC.**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).

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

Returns: dataframe with subjects removed that exceed cutoff value.

methylcheck.postprocessQC.**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.

methylcheck.postprocessQC.**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.

silent: suppresses figure, so no output unless save==True too.

methylcheck.postprocessQC.**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).

CHAPTER 5

Indices and tables

- genindex
- modindex
- search

Python Module Index

m

Index