

Use for GMM

The following import statement is required to avoid a warning when we use the GMM. There is a memory leak when we use GMM. Do not use it for other entropy approximations because it slows down the campaign in general.

Note: it causes a problem with the plots. They are created twice.

```
In [1]: #import os
#os.environ["OMP_NUM_THREADS"] = '1'
```

```
In [2]: import MyPlotting as MyPlots
import matplotlib.pyplot as plt
from datastruct import Settings, Experiment, ExperimentStep, DataPoint
from entropy import calc_entropy, default_entropy_options
import numpy as np

from functions_gain_factor import recalculating_entropy
```

9 Reloading Experiments

```
In [3]: # #Reloading Data
exp_control = Experiment.load('11/test1_control_id11c.gz', recalc_yprofs=True)
exp_entropy1 = Experiment.load('11/test1_entropy_id11e1.gz', recalc_yprofs=True)
exp_entropy2 = Experiment.load('11/test1_entropy_id11e2.gz', recalc_yprofs=True)
exp_on_the_fly1 = Experiment.load('11/test1_on_the_fly1_id11o1.gz', recalc_yprofs=True)
exp_on_the_fly2 = Experiment.load('11/test1_on_the_fly2_id11o2.gz', recalc_yprofs=True)

#####
#####Use only when you have a GMM version of
####GMM case
# exp_control_gmm = Experiment.Load('1/test1_control_id1c_gmm.gz', recalc_yprofs=True)
# exp_entropy1_gmm = Experiment.Load('1/test1_entropy_id1e1_gmm.gz', recalc_yprofs=True)
# exp_entropy2_gmm = Experiment.Load('1/test1_entropy_id1e2_gmm.gz', recalc_yprofs=True)
# exp_on_the_fly1_gmm = Experiment.Load('1/test1_on_the_fly1_id1o1_gmm.gz', recalc_yprofs=True)
# exp_on_the_fly2_gmm = Experiment.Load('1/test1_on_the_fly2_id1o2_gmm.gz', recalc_yprofs=True)

#reloaded_exp.creation_time.isoformat()
```

```
In [ ]:
```

```
In [4]: #Notice now all the outcomes have the same size except for the datax, datay and datady
#Last unfitted point
len(exp_entropy1.entropy()) #34
len(exp_entropy1.load_yprofs()) #34
len(exp_entropy1.load_pts()) #34
len(exp_entropy1.getdata()[0]) #35
len(exp_entropy1.meastimes()) #34
```

```
C:\Users\rober\FINAL NIST PROJECT\datastruct.py:346: VisibleDeprecationWarning: Creating an ndarray from ragged nested sequences (which is a list-or-tuple of lists-or-tuples-or ndarrays with different lengths or shapes) is deprecated. If you meant to do this, you must specify 'dtype=object' when creating the ndarray.
    return np.array(all_yprof)
C:\Users\rober\FINAL NIST PROJECT\datastruct.py:353: VisibleDeprecationWarning: Creating an ndarray from ragged nested sequences (which is a list-or-tuple of lists-or-tuples-or ndarrays with different lengths or shapes) is deprecated. If you meant to do this, you must specify 'dtype=object' when creating the ndarray.
    return np.array(all_pts)
```

Out[4]: 15

Warning

Remember that the maximum time spend in the control data is way greater than for the autonomous cases. Thus, when we are plotting the y-profiles and the histogram for the parameters we should may be truncated the values of control until a time that is closer to max amount of time spend in the autonomous cases.

10 Compare to Benchmark

```
In [5]: # from matplotlib.colors import LogNorm
# from numpy.matlib import repmat

print(exp_entropy1.settings.ground_truth_pars) #{'I0': 30.0, 'A': 15.0, 'phi0': 155.0,
print(exp_on_the_fly1.settings.ground_truth_pars)

{'I0': 300.0, 'A': 150.0, 'phi0': 155.0, 'T': 401.0, 'sigma': 797.0}
{'I0': 300.0, 'A': 150.0, 'phi0': 155.0, 'T': 401.0, 'sigma': 797.0}
```

10.1 Y-Profiles Histograms

```
In [6]: ##Plotting the last set of Y-profiles for the different approaches
##If you do not want to plot the figure of merits do not provide them. It still works

#Entropy 1
MyPlots.plot_y_profiles(exp_entropy1.load_yprofs()[-1],exp_entropy1.settings.x,
                      FOM = exp_entropy1.load_FOM()[-1],
                      this_title = "Histogram of Y profiles Entropy 1 (Selected) Approach")

#Entropy 2
MyPlots.plot_y_profiles(exp_entropy2.load_yprofs()[-1],exp_entropy2.settings.x,
                      FOM = exp_entropy2.load_FOM()[-1],
                      this_title = "Histogram of Y profiles Entropy 2 (All) Approach")

#On-the Fly 1
MyPlots.plot_y_profiles(exp_on_the_fly1.load_yprofs()[-1],exp_on_the_fly1.settings.x,
                      FOM = exp_on_the_fly1.load_FOM()[-1],
                      this_title = "Histogram of Y profiles On-The_Fly 1 Approach")

#On-the Fly 1
MyPlots.plot_y_profiles(exp_on_the_fly2.load_yprofs()[-1],exp_on_the_fly2.settings.x,
                      FOM = exp_on_the_fly2.load_FOM()[-1],
                      this_title = "Histogram of Y profiles On-The_Fly 2 Approach")

#Control Data
```

```
MyPlots.plot_y_profiles(exp_control.load_yprofs()[-1],exp_control.settings.x,  
this_title = "Histogram of Y profiles My Control Evenly Approa
```

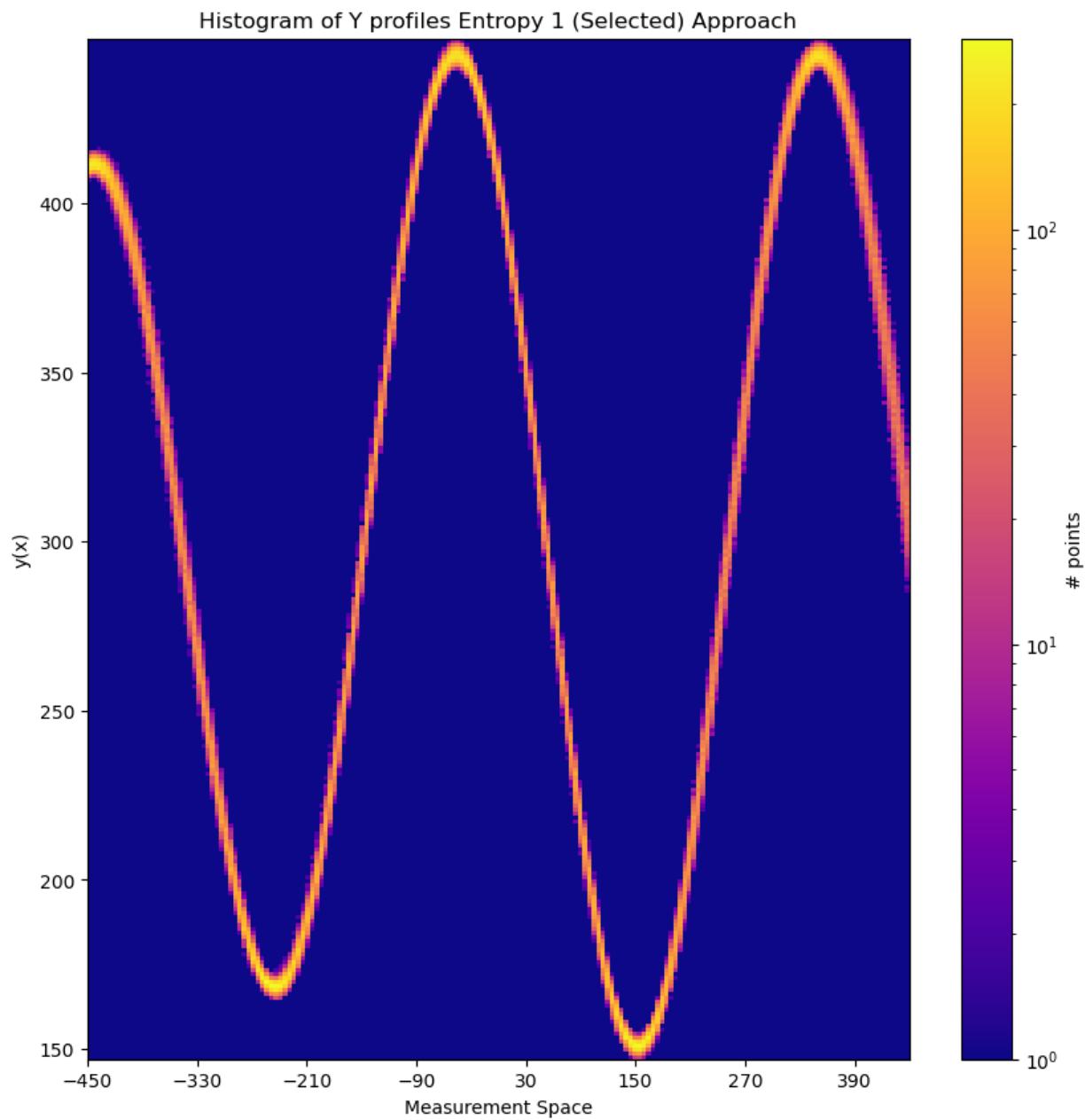
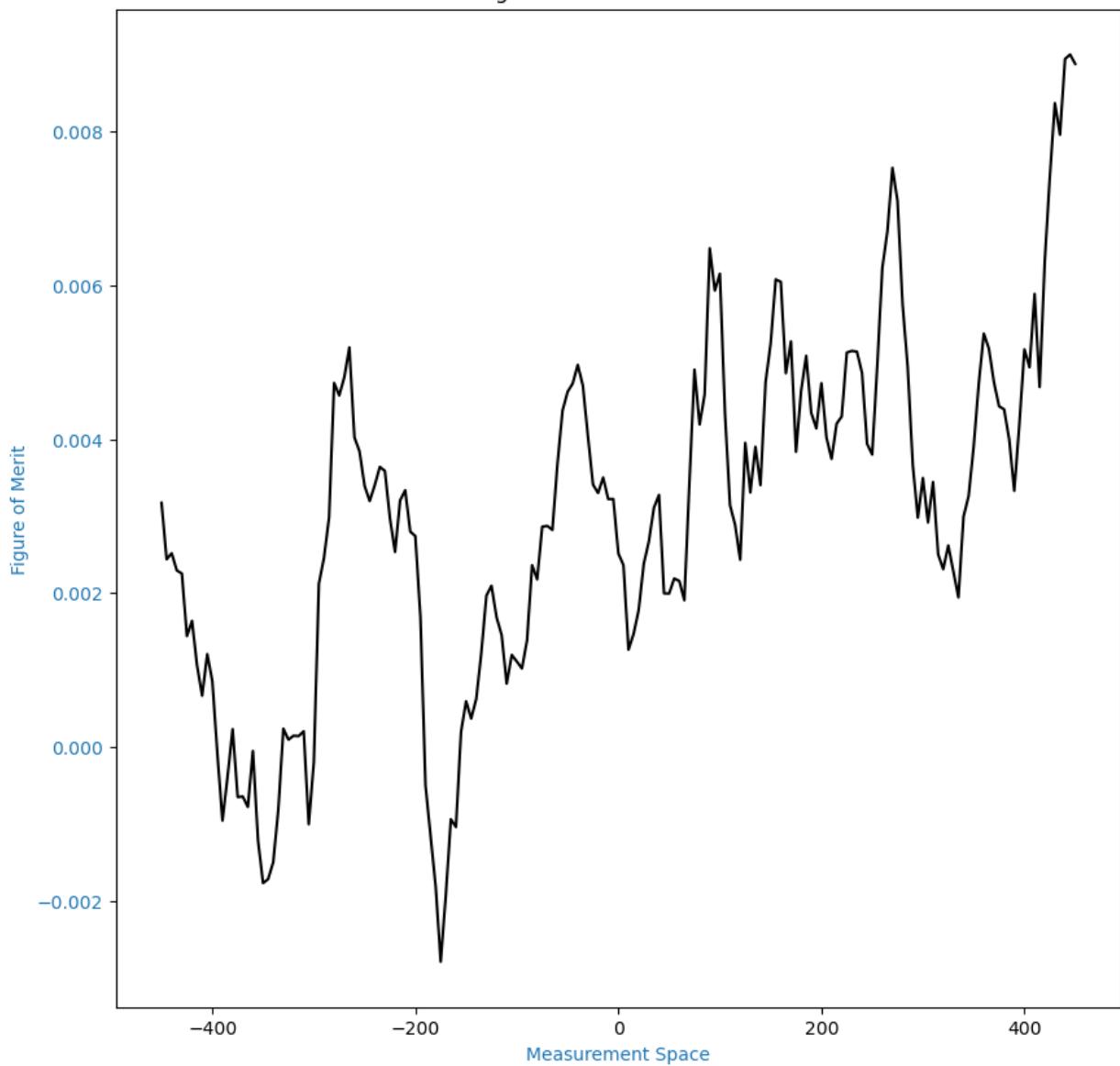


Figure of Merits for each x



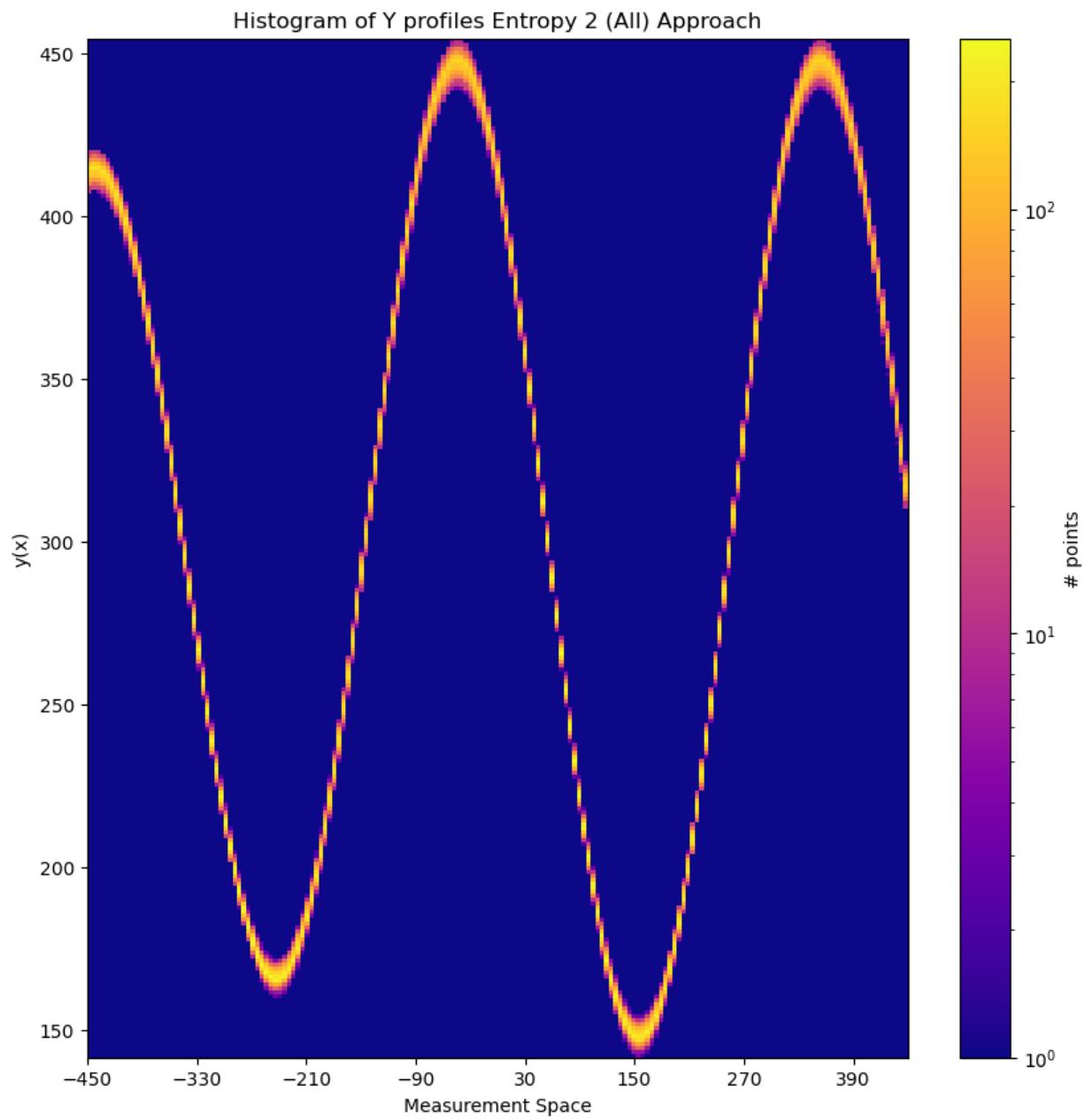
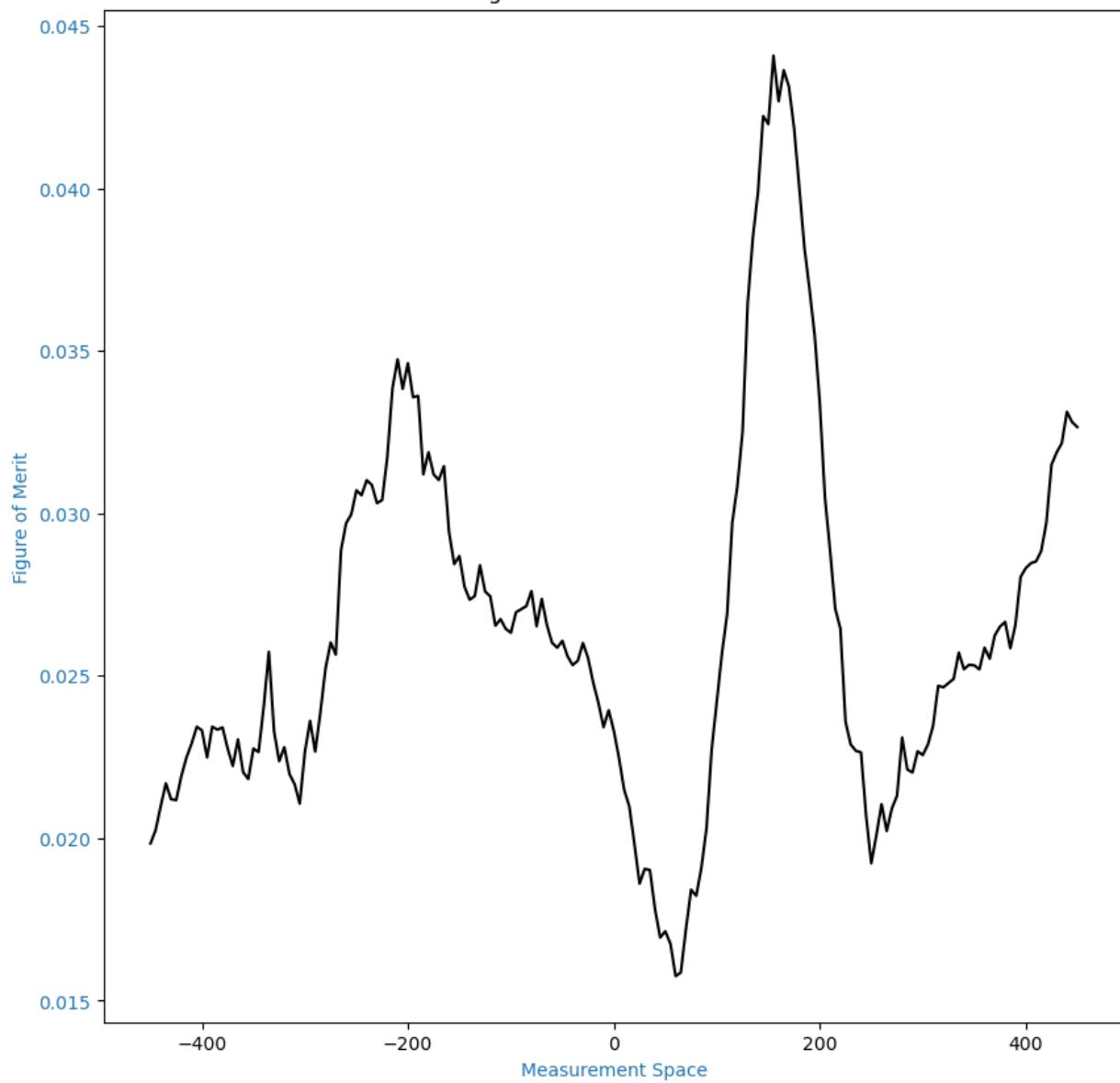


Figure of Merits for each x



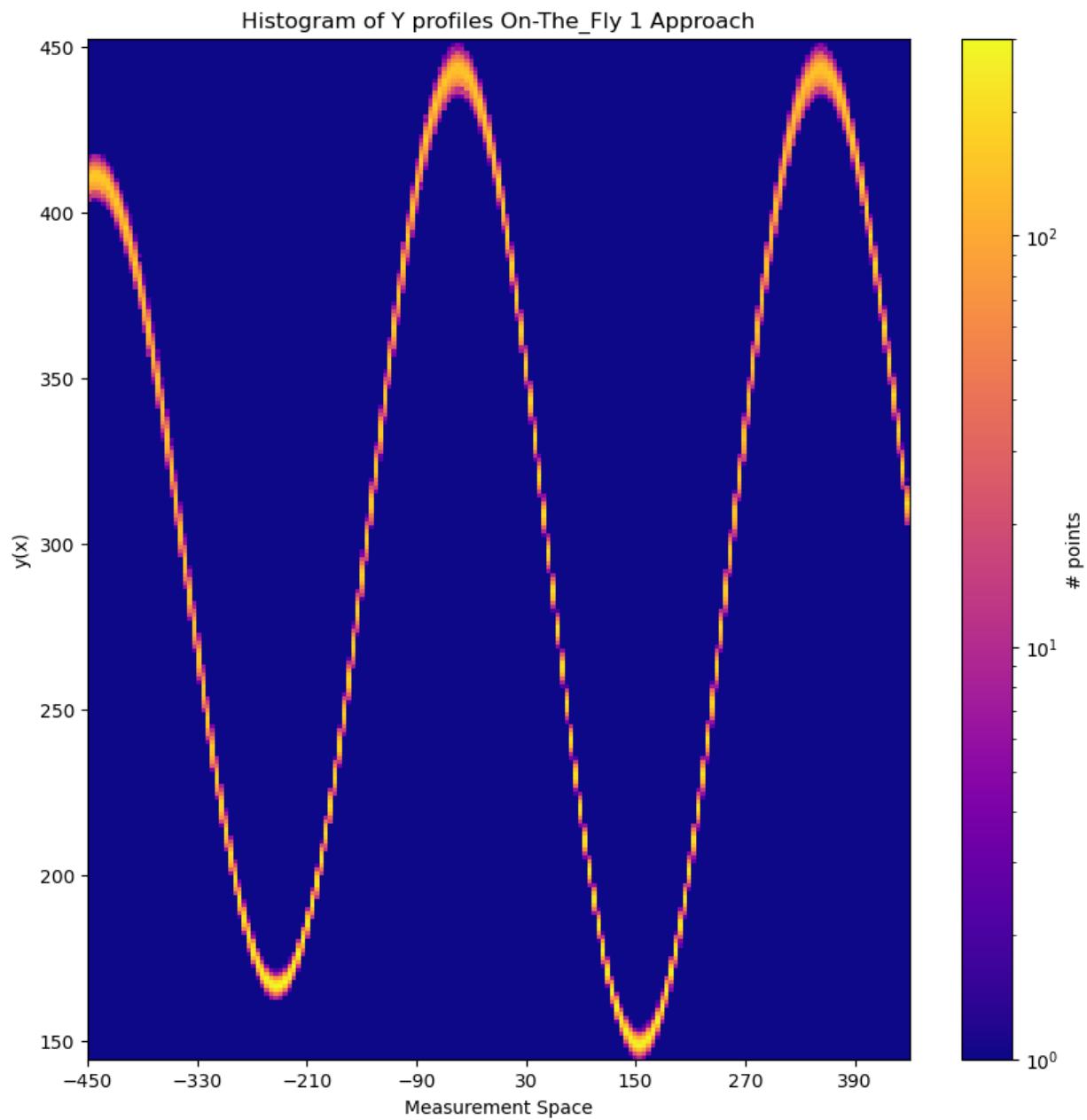
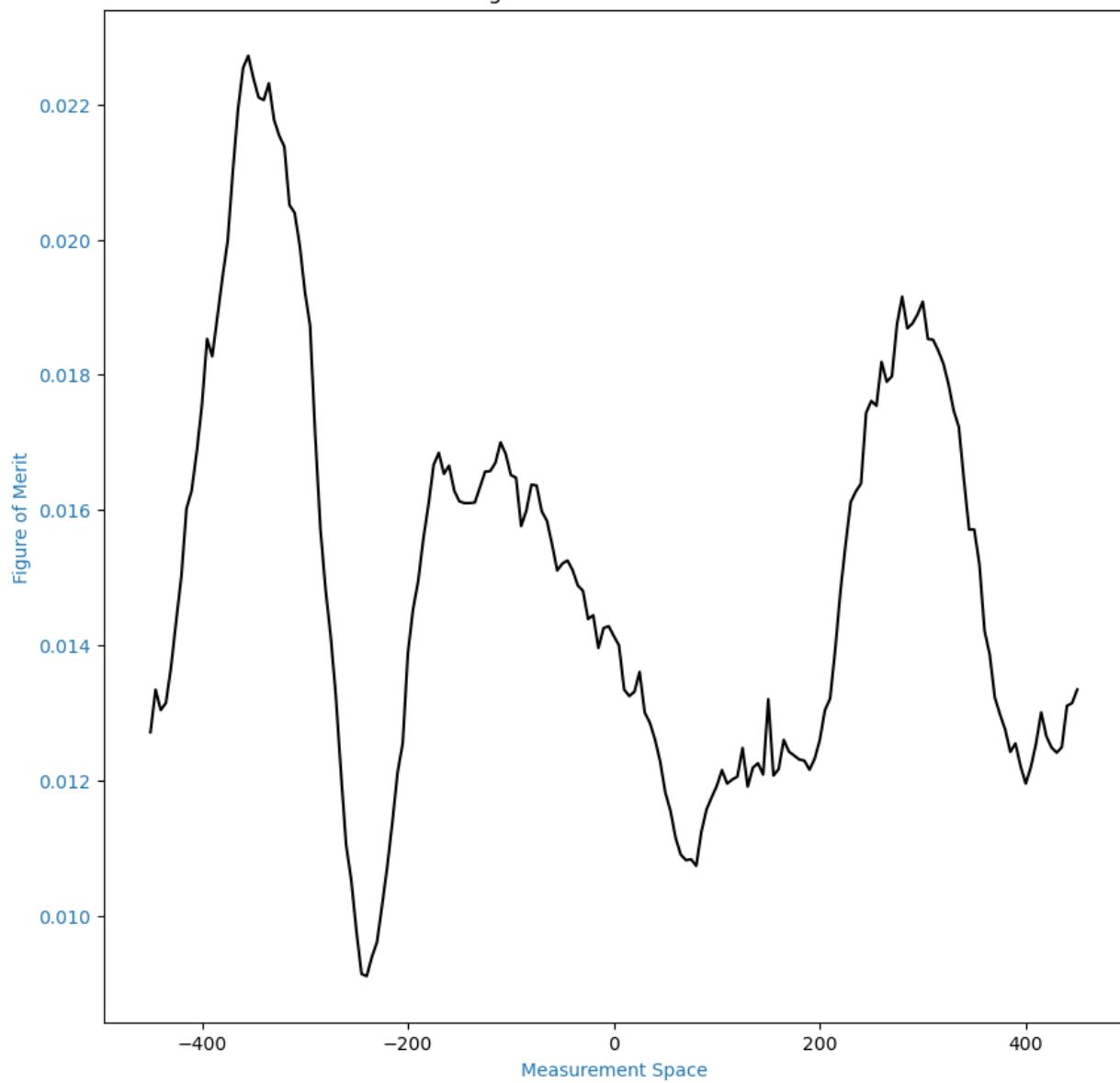


Figure of Merits for each x



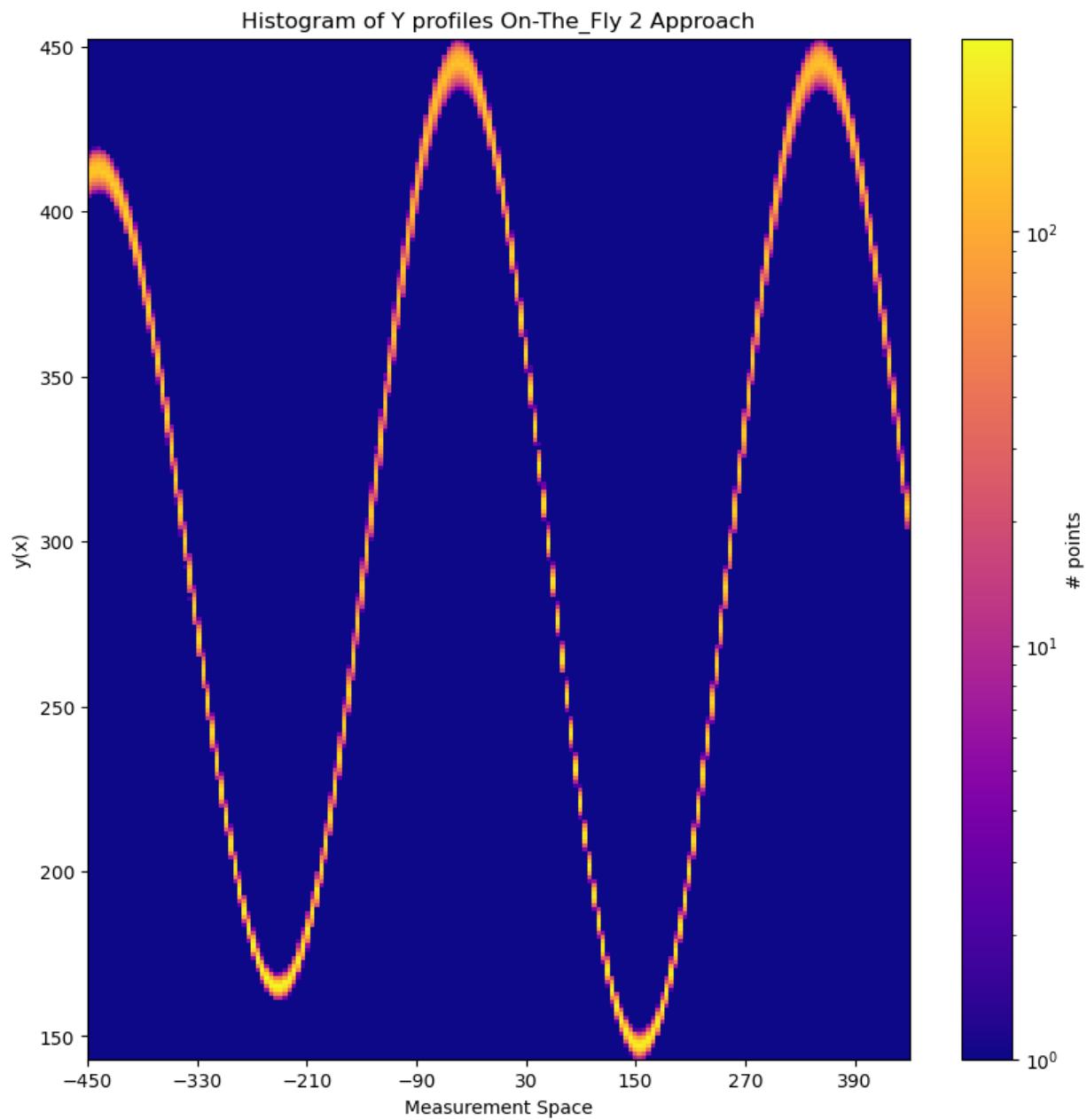
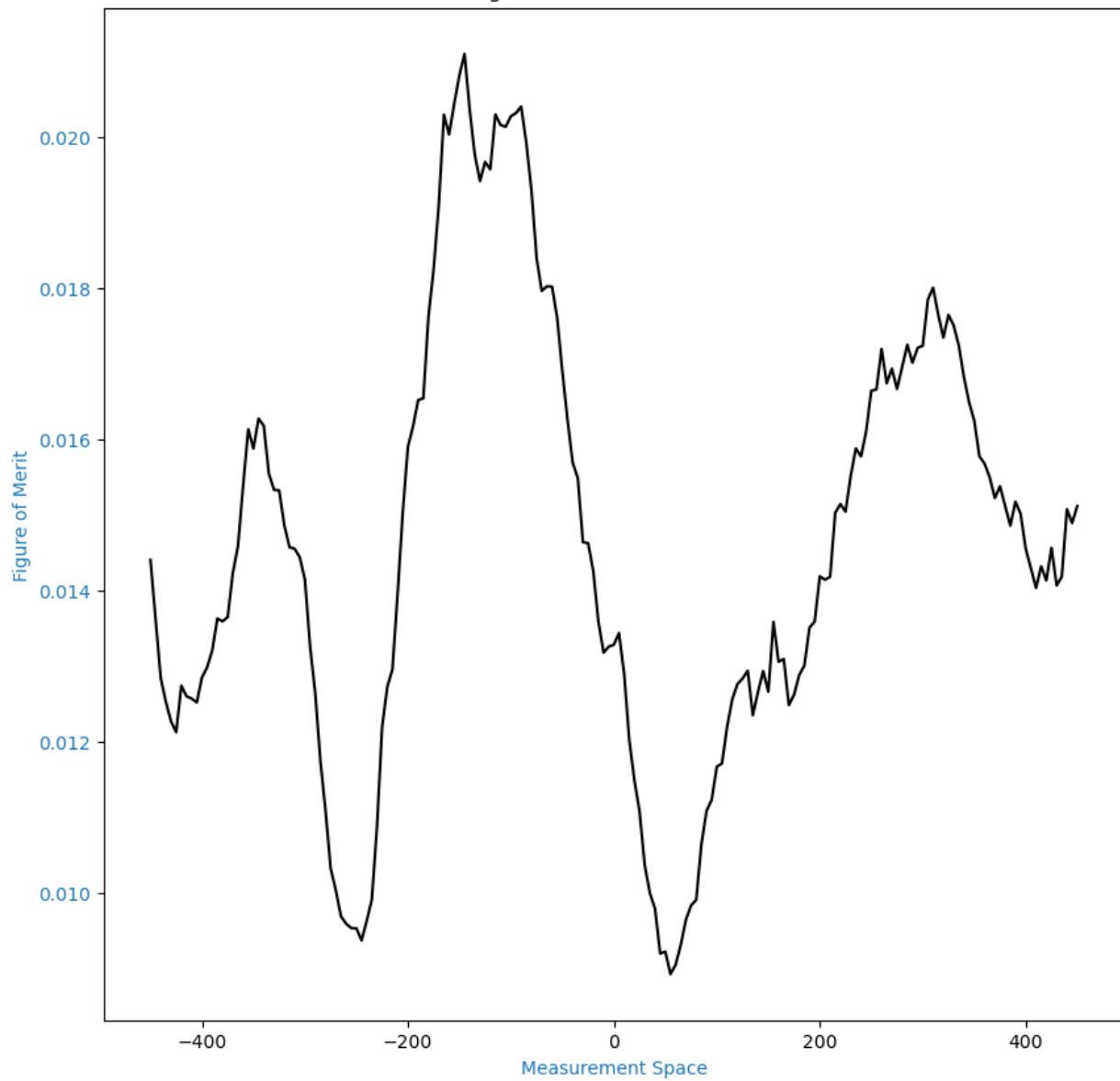
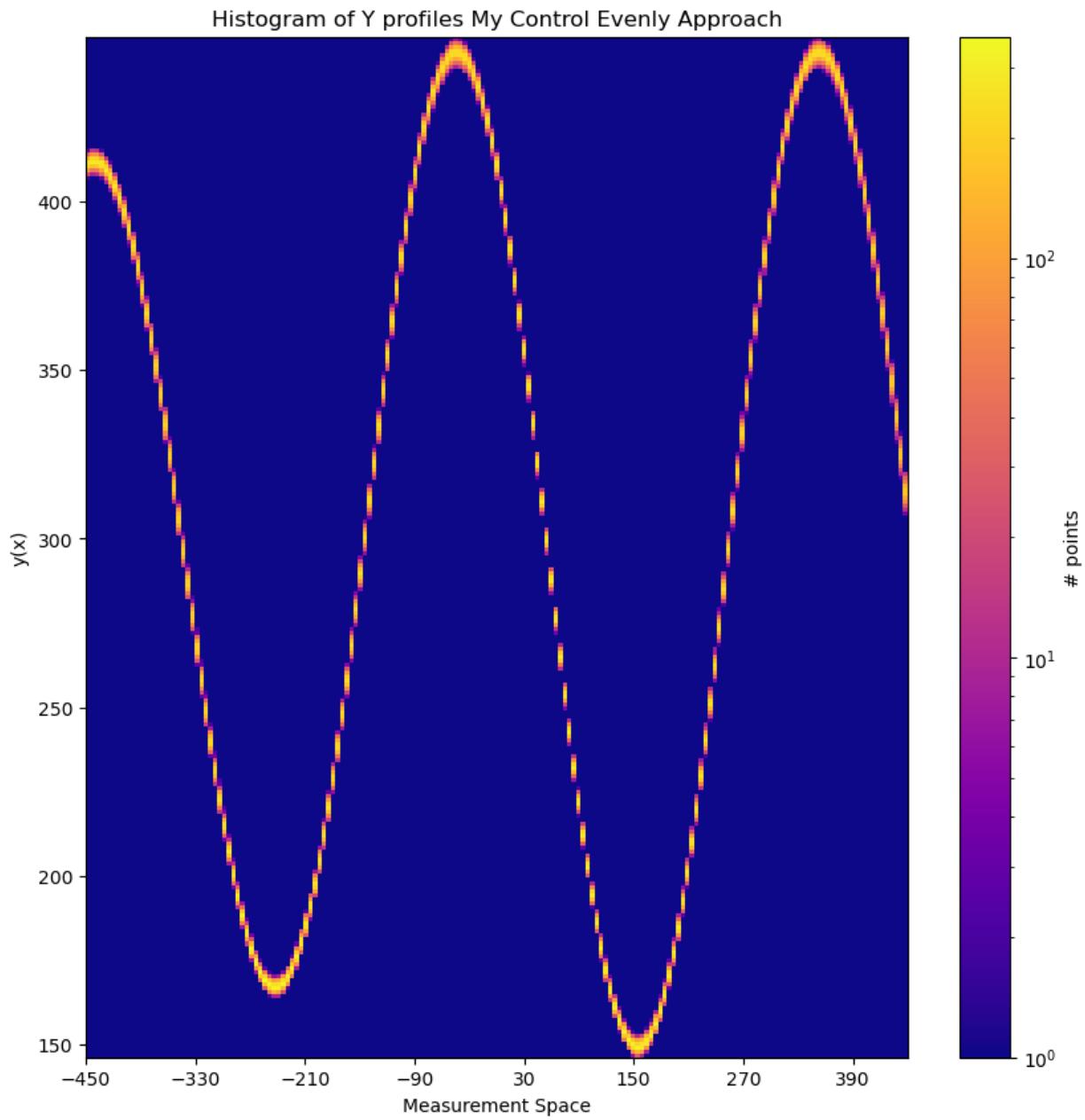


Figure of Merits for each x





In [7]: ##### GMM TO COMPARE Commnet out everything below unless you have a reason

```
# #Entropy 1
# MyPlots.plot_y_profiles(exp_entropy1_gmm.Load_yprofs()[-1],exp_entropy1_gmm.settings,
# FOM = exp_entropy1_gmm.load_FOM()[-1],
# this_title = "Histogram of Y profiles Entropy 1 (Selected) GMM Approach")

# #Entropy 2
# MyPlots.plot_y_profiles(exp_entropy2_gmm.Load_yprofs()[-1],exp_entropy2_gmm.settings,
# FOM = exp_entropy2_gmm.load_FOM()[-1],
# this_title = "Histogram of Y profiles Entropy 2 (ALL) GMM Approach")

# #On-the Fly 1
# MyPlots.plot_y_profiles(exp_on_the_fly1_gmm.Load_yprofs()[-1],exp_on_the_fly1_gmm.settings,
# FOM = exp_on_the_fly1_gmm.load_FOM()[-1],
# this_title = "Histogram of Y profiles On-The_Fly 1 GMM Approach")

# #On-the Fly 1
# MyPlots.plot_y_profiles(exp_on_the_fly2_gmm.Load_yprofs()[-1],exp_on_the_fly2_gmm.settings,
# FOM = exp_on_the_fly2_gmm.load_FOM()[-1],
```

```
# this_title = "Histogram of Y profiles On-The_Fly 2 GMM Approach"
# #Control Data
# MyPlots.plot_y_profiles(exp_control_gmm.load_yprofs()[-1],exp_control_gmm.settings.x)
# this_title = "Histogram of Y profiles My Control Evenly GMM"
```

10.2 Histogram for the Parameters Samples

Getting the Lists for the Parameters The containers total_pts (and similars) stores 2-D arrays on them; each 2-d array represents an iteration. The 2-d array contain the samples for all the parameters n_samplesn_parameters (*where the number of samples can change per iteration, but the number of columns is constant*). First,based on the columns of each array (each column represents a parameter) we will create lists that represent each parameter where each sub-list inside the list for a given paremeter represents the samples at an iterations, n_iterationsn_samples

Note: 1) Any of the 2-d arays in this_total_pts and this_total_pts2 have as columns representing each parameter. The columns represent respectvily A, I0, T, and phi0. They are in the same order than in the console output from the loop.

2) We Cannot create 2D arrays for each parameter because the number of samples per iteration can change. Thus, we create lists for each parameter.

plottinh_hist() Here we create an auxiliary funtion to plot each of the sublists (that represent the samples at all the iterations for a given parameter) of the output of using the function above.

WARNING: Remember that the number of samples per iteration can change. Thus, a histogram may not be the best way to visualize converge. we have two options.

1. Normalize the number of samples for al iterations.
2. commnet out the line of code mark_outliers() in the main loop. So, we do not rule outliers and then we will have the same number of samples for all iterations.

plottinh_hist()

In [8]: *#getting the Lists for both approaches. The lists contains lists where each sublists is sublists that represent a parameter there are sublists that represent the samples for #at an iteration.*

```
#Entropy Method
list_par_separated_e1 = MyPlots.getting_list_each_par(exp_entropy1.load_pts()) #This line
list_par_separated_e2 = MyPlots.getting_list_each_par(exp_entropy2.load_pts())
#On-the-fly 1 Method
list_par_separated_o1 = MyPlots.getting_list_each_par(exp_on_the_fly1.load_pts())
#On-the-fly 2 Method
list_par_separated_o2 = MyPlots.getting_list_each_par(exp_on_the_fly2.load_pts())
#Control method
list_par_separated_c = MyPlots.getting_list_each_par(exp_control.load_pts())
#####
#####GMM VERSION
#
# #Entropy Method
```

```
# list_par_separated_e1_gmm = MyPlots.getting_list_each_par(exp_entropy1_gmm.Load_pts())
# list_par_separated_e2_gmm = MyPlots.getting_list_each_par(exp_entropy2_gmm.Load_pts())
# #On-the-fly 1 Method
# list_par_separated_o1_gmm = MyPlots.getting_list_each_par(exp_on_the_fly1_gmm.Load_pts())
# #On-the-fly 2 Method
# list_par_separated_o2_gmm = MyPlots.getting_list_each_par(exp_on_the_fly2_gmm.Load_pts())
# #Control method
# list_par_separated_c_gmm = MyPlots.getting_list_each_par(exp_control_gmm.Load_pts())
```

```
In [9]: aux_list = [list_par_separated_e1[0], list_par_separated_e2[0], list_par_separated_o1[0],
               list_par_separated_o2[0], list_par_separated_c[0]]

list_times = [exp_entropy1.totaltimes(), exp_entropy2.totaltimes(), exp_on_the_fly1.totaltimes(),
              exp_on_the_fly2.totaltimes(), exp_control.totaltimes()]

names_for_approaches = ["Entropy 1 (Selected) MVN", "Entropy 2 (All) MVN", "On-the fly",
                        "Control-45 even MVN"]
par_name = "A"

# specify y-edges for all three histograms
y_min, y_max = MyPlots.finding_max_min(aux_list)

for i in range(len(aux_list)):
    MyPlots.plotting_hist_logtime(aux_list[i], names_for_approaches[i], par_name, y_min, y_max)

#####
#####GMM VERSION COMMENT OUT IF YOU DO NOT HAVE A GMM VERSION

# aux_list = [list_par_separated_e1_gmm[0], list_par_separated_e2_gmm[0], list_par_separated_o1_gmm[0],
#             list_par_separated_o2_gmm[0], list_par_separated_c_gmm[0]]

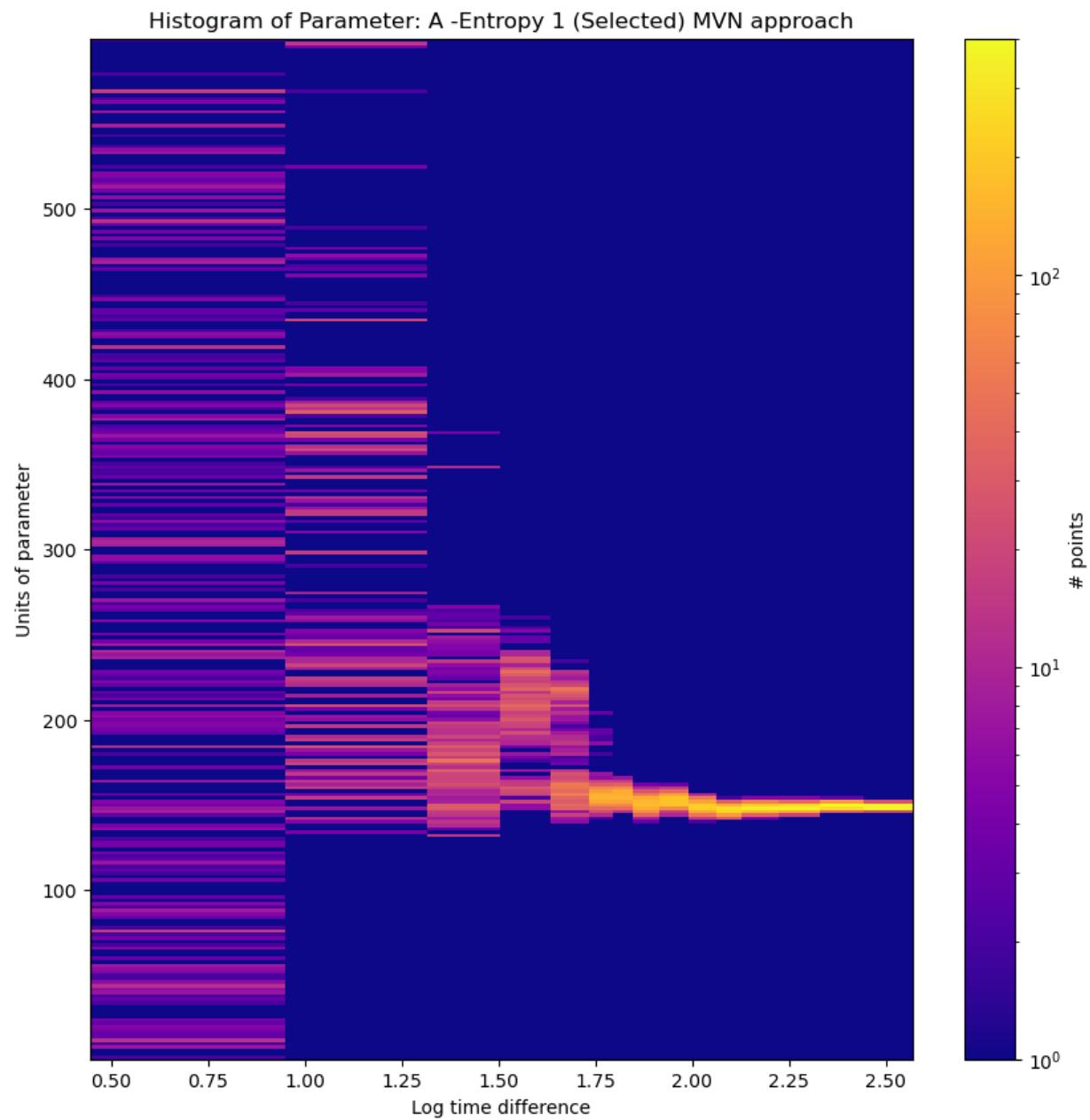
# names_for_approaches = ["Entropy 1 (Selected) GMM", "Entropy 2 (ALL) GMM", "On-the fly",
#                         "Control-45 even GMM"]
# par_name = "A"

# # specify y-edges for all three histograms
# y_min, y_max = MyPlots.finding_max_min(aux_list)

# for i in range(len(aux_list)):
#     MyPlots.plotting_hist(aux_list[i], names_for_approaches[i], par_name, y_min, y_max)
```

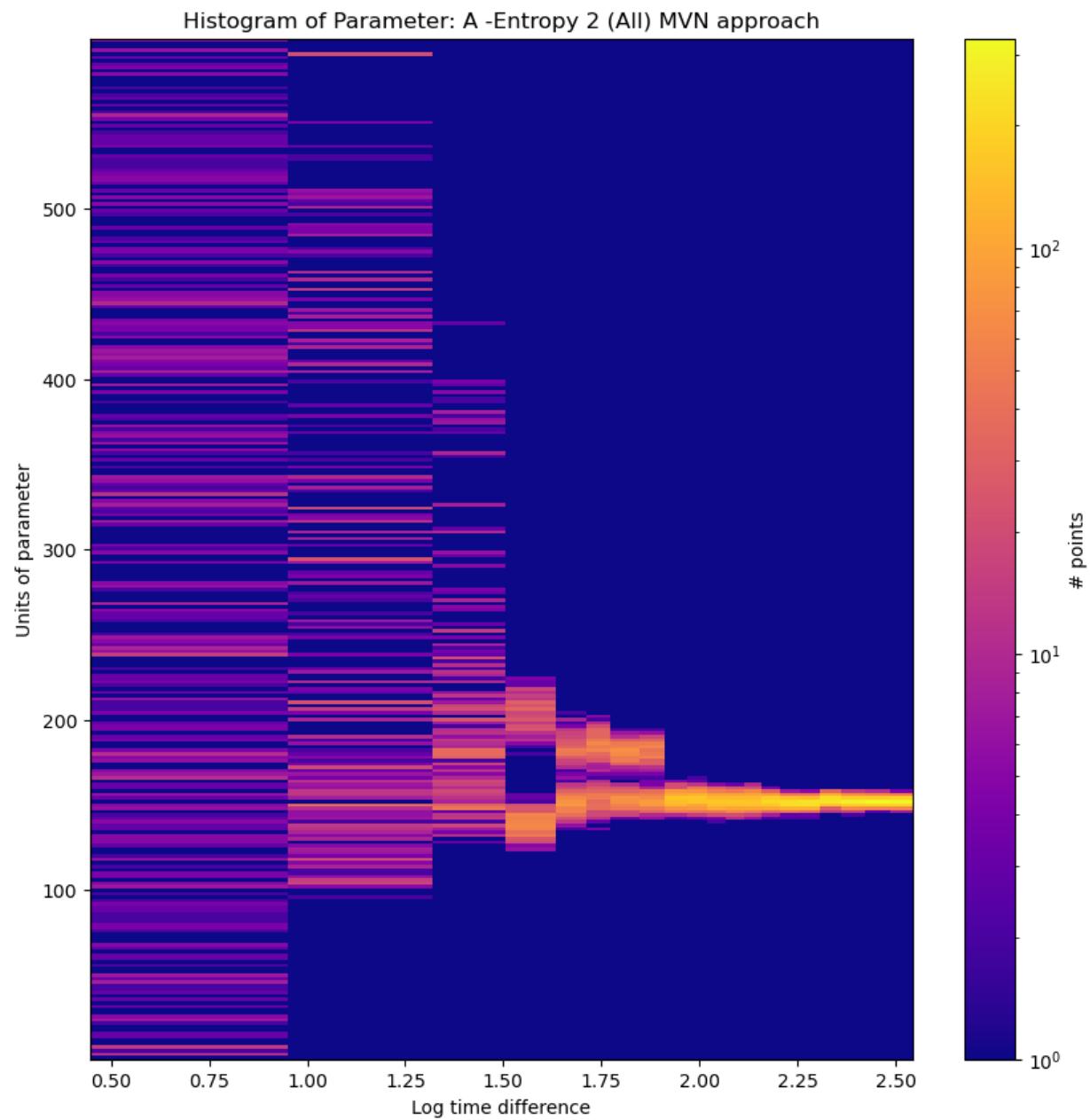
my x edges

```
[0.44639502 0.95154499 1.31774187 1.50557368 1.63498984 1.73417367
 1.7955323 1.84686347 1.91483462 1.98918063 2.06119796 2.12836379
 2.22187384 2.32862824 2.44108604 2.56856595]
```



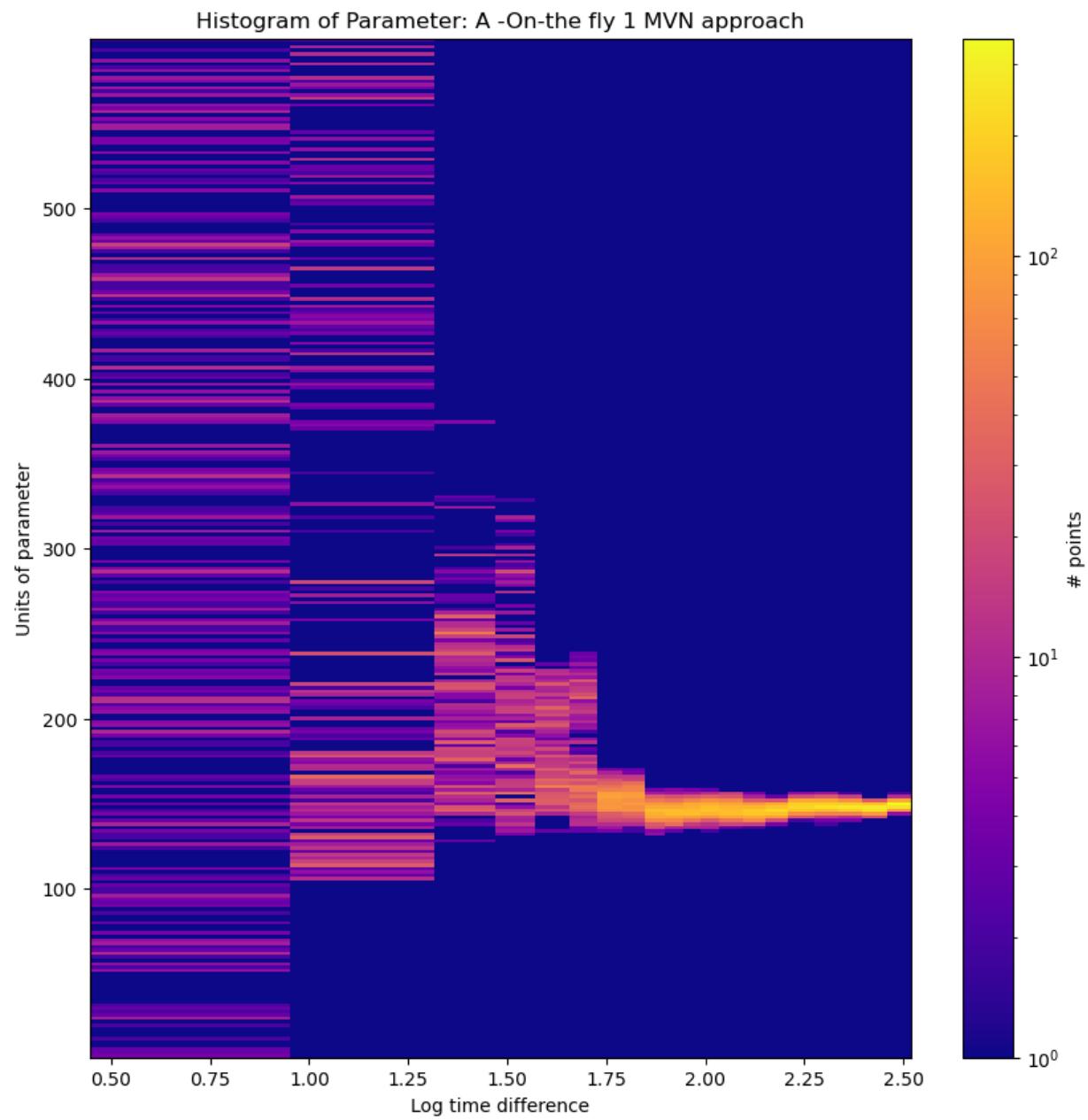
my_x_edges

```
[0.44639502 0.95154499 1.31774187 1.50557368 1.63498984 1.71129492
 1.77265356 1.84686347 1.91016642 1.96791968 2.01978708 2.06722129
 2.11291436 2.15680688 2.2041022 2.25523444 2.30786839 2.36212059
 2.4234927 2.48599068 2.54348622]
```



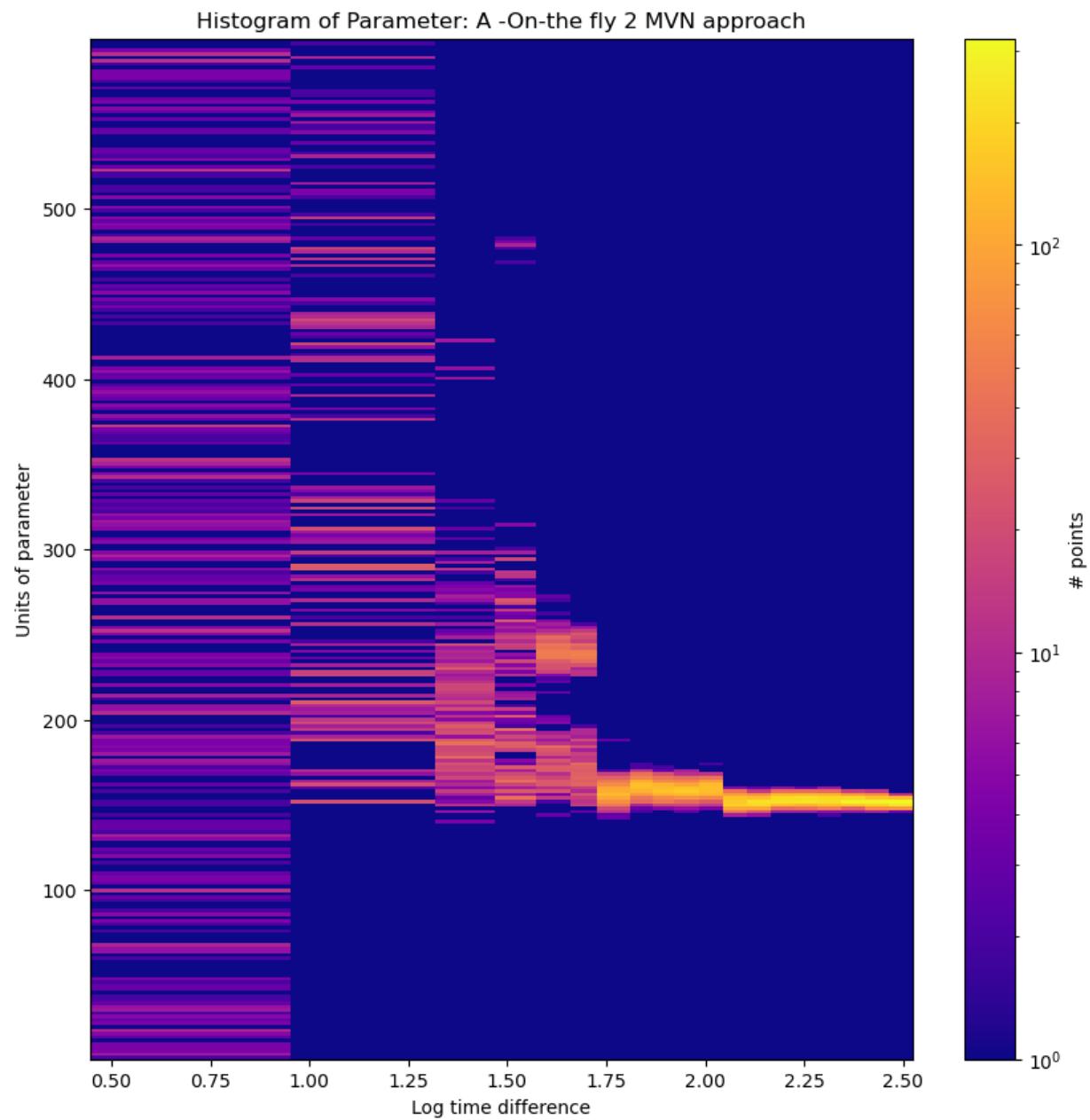
my_x_edges

```
[0.44639502 0.95154499 1.31774187 1.46825687 1.56930922 1.65735485
 1.72604662 1.79184115 1.84678641 1.89944205 1.9452213 1.98788317
 2.03501492 2.09497142 2.15098144 2.2097724 2.27432117 2.33697783
 2.39689737 2.45818412 2.51950998]
```

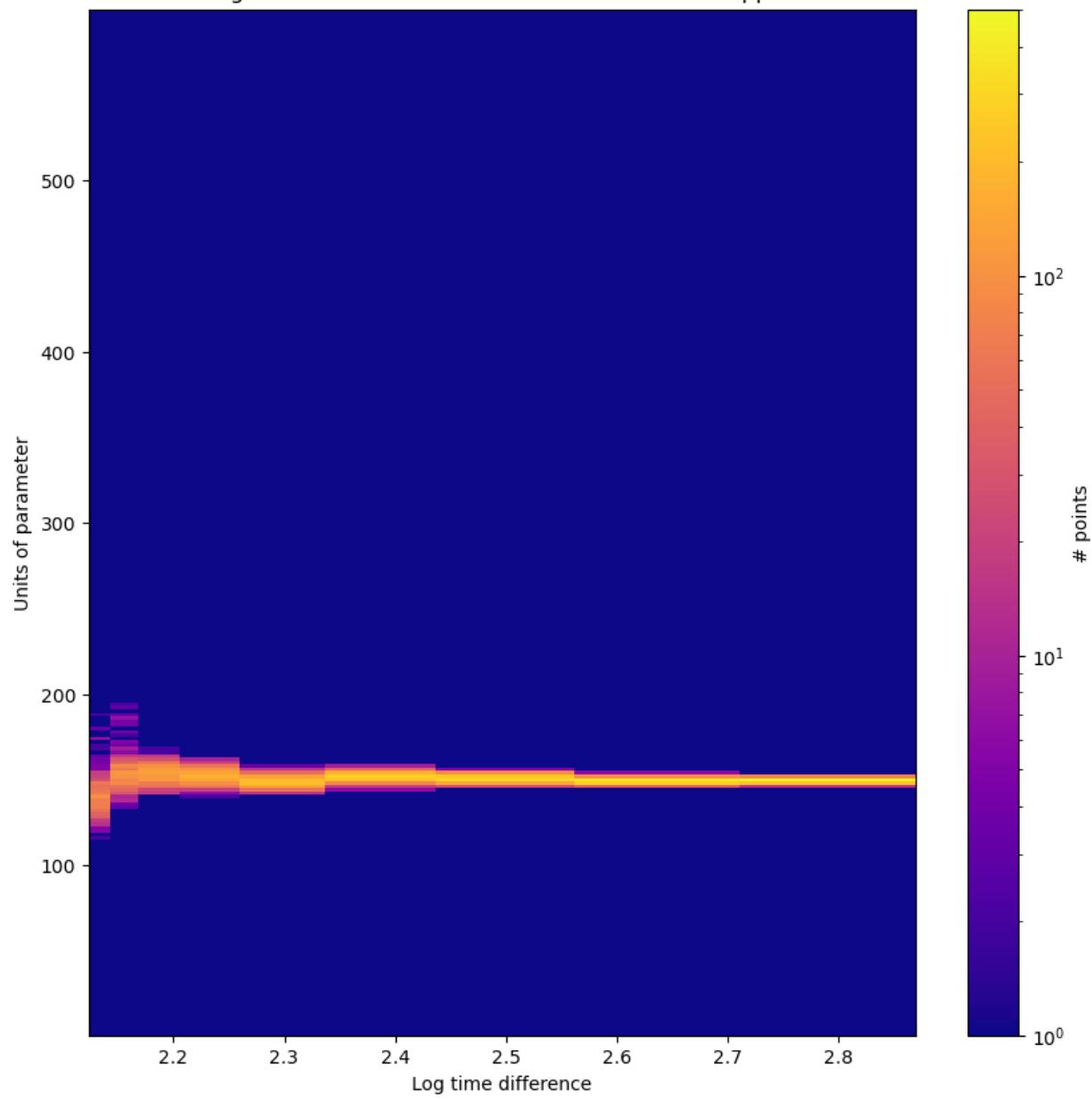


my x edges

```
[0.44639502 0.95154499 1.31774187 1.46911739 1.57081045 1.65856962
 1.72708783 1.80883598 1.8676842 1.92186855 1.98561851 2.04632984
 2.10560611 2.16596571 2.22427618 2.28267561 2.34410219 2.40478666
 2.46419209 2.52344831]
```



Histogram of Parameter: A -Control-45 even MVN approach



```
In [10]: aux_list = [list_par_separated_e1[1], list_par_separated_e2[1], list_par_separated_o1[1],
               list_par_separated_o2[1], list_par_separated_c[1]]

list_times = [exp_entropy1.totaltimes(), exp_entropy2.totaltimes(), exp_on_the_fly1.to,
              exp_on_the_fly2.totaltimes(), exp_control.totaltimes()]

names_for_approaches = ["Entropy 1 (Selected)", "Entropy 2 (All)", "On-the fly 1", "On-t
                        "Control-45 even"]

par_name = "I0"

# specify y-edges for all three histograms

#WARNING: If you are plotting the GMM versions, you will need to include the values us
#values below
y_min, y_max = MyPlots.finding_max_min(aux_list)
```

```
for i in range(len(aux_list)):
    MyPlots.plotting_hist_logtime(aux_list[i], names_for_approaches[i], par_name, y_mi

#####
#####GMM VERSION COMMENT OUT IF YOU DO NOT HAVE A GMM VERSION

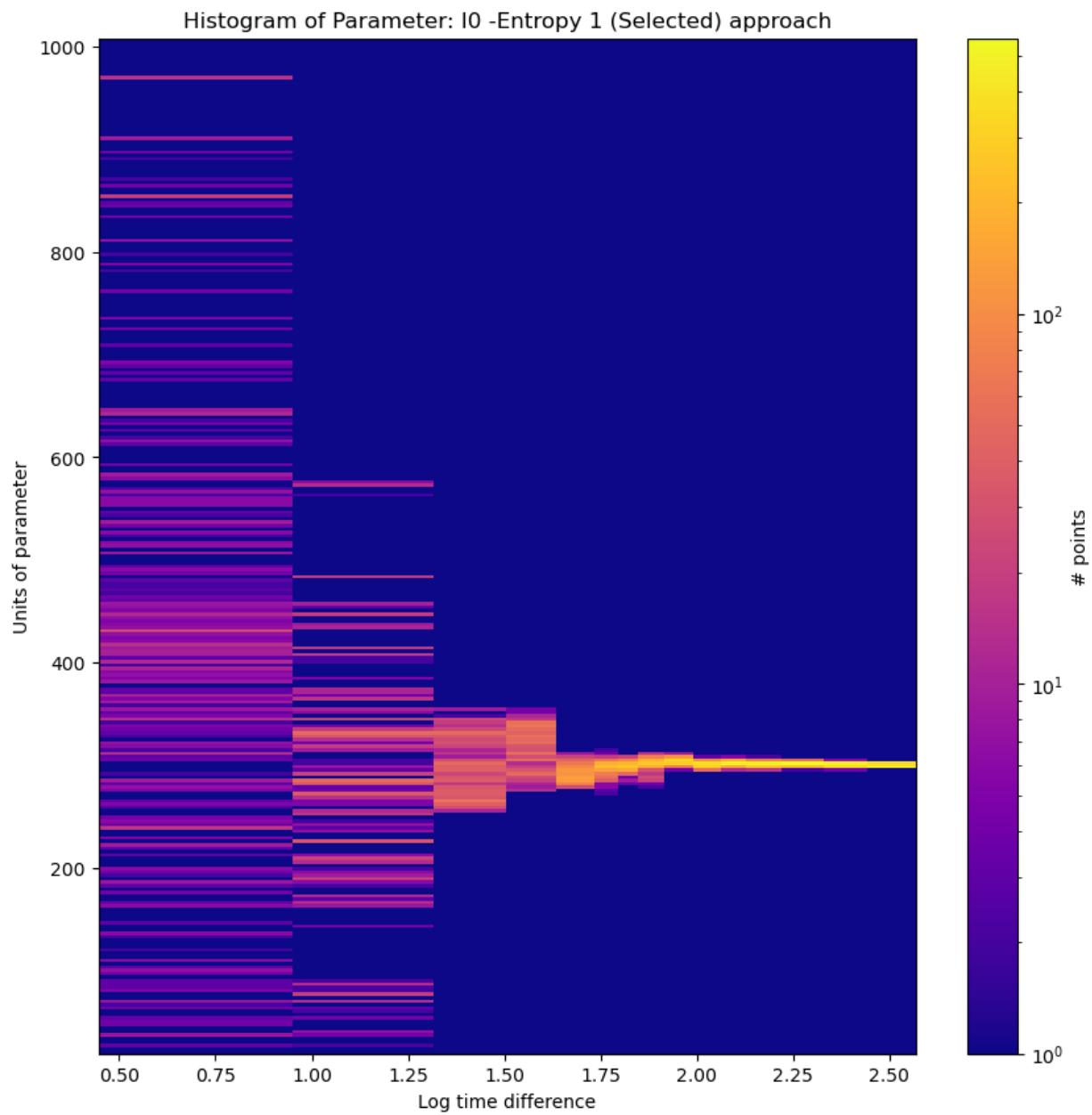
# aux_list = [list_par_separated_e1_gmm[1], list_par_separated_e2_gmm[1], list_par_sep
#             list_par_separated_o2_gmm[1], list_par_separated_c_gmm[1]]

# names_for_approaches = ["Entropy 1 (Selected) GMM", "Entropy 2 (ALL) GMM", "On-the fl
#                         "Control-45 even GMM"]
# par_name = "I0"

#
# # specify y-edges for all three histograms
# y_min, y_max = MyPlots.finding_max_min(aux_list)

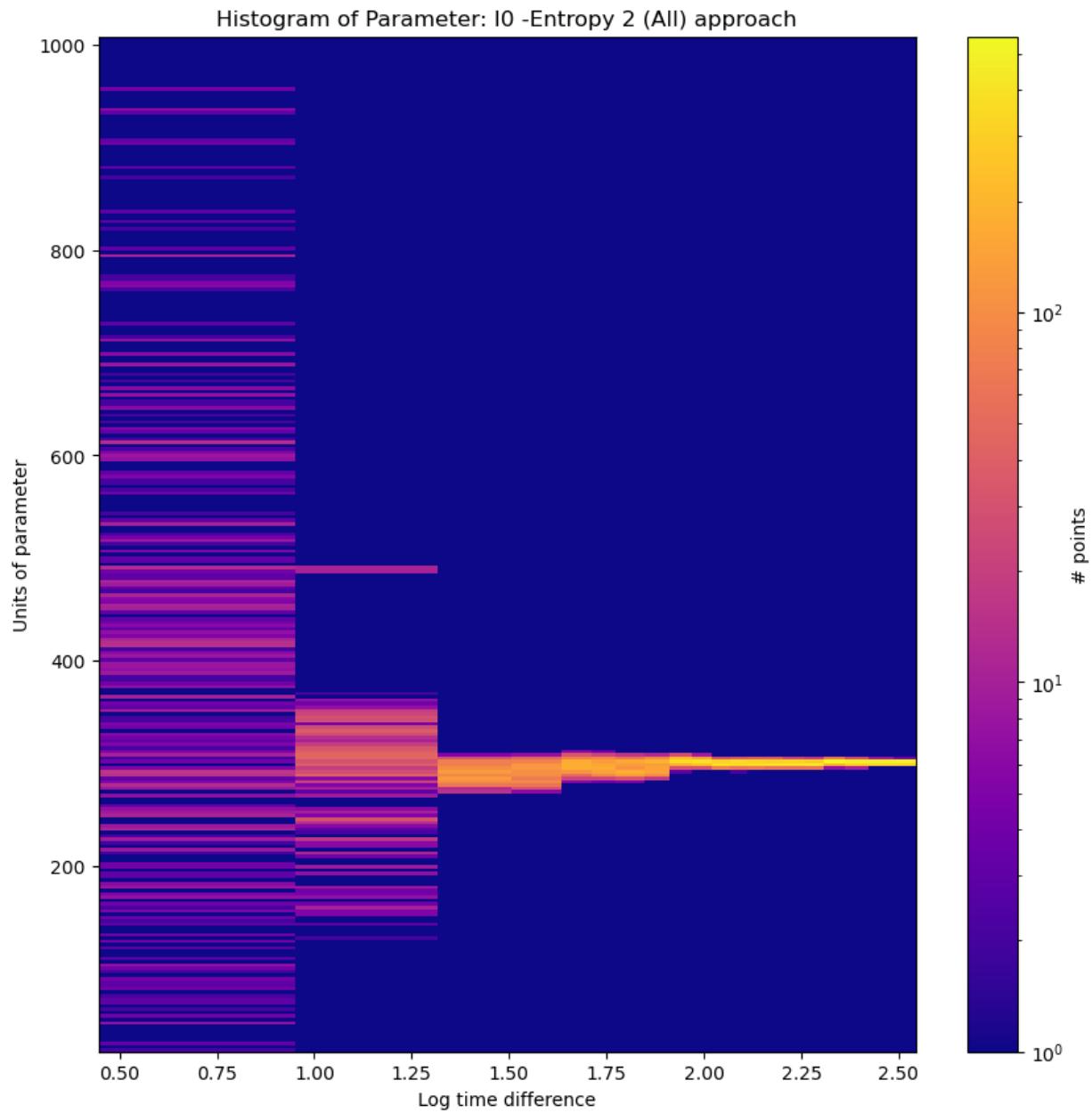
# for i in range(len(aux_list)):
#     MyPlots.plotting_hist(aux_list[i], names_for_approaches[i], par_name, y_min, y_n

my x edges
[0.44639502 0.95154499 1.31774187 1.50557368 1.63498984 1.73417367
 1.7955323 1.84686347 1.91483462 1.98918063 2.06119796 2.12836379
 2.22187384 2.32862824 2.44108604 2.56856595]
```



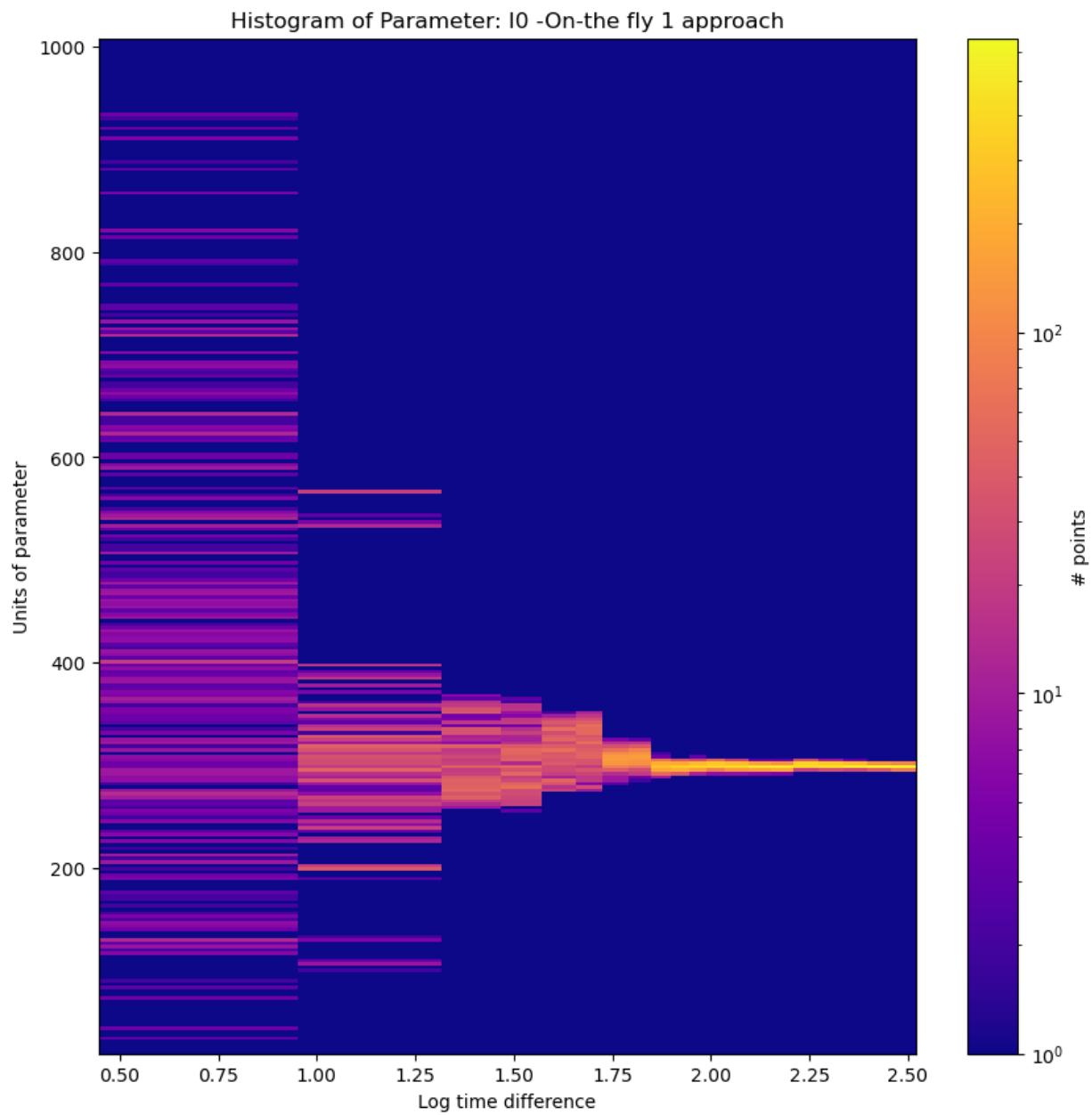
my x edges

```
[0.44639502 0.95154499 1.31774187 1.50557368 1.63498984 1.71129492
 1.77265356 1.84686347 1.91016642 1.96791968 2.01978708 2.06722129
 2.11291436 2.15680688 2.2041022 2.25523444 2.30786839 2.36212059
 2.4234927 2.48599068 2.54348622]
```



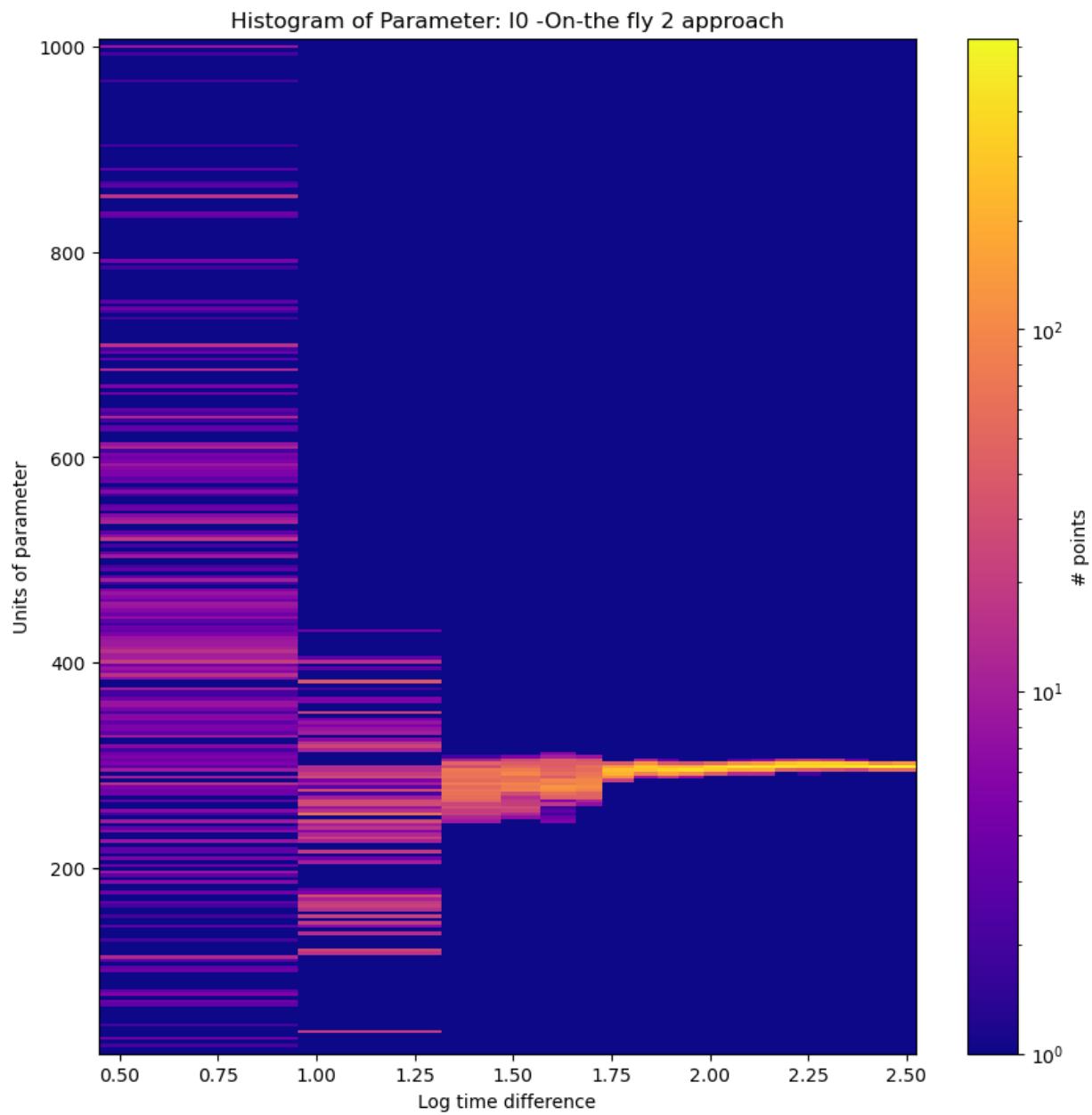
my x edges

```
[0.44639502 0.95154499 1.31774187 1.46825687 1.56930922 1.65735485
 1.72604662 1.79184115 1.84678641 1.89944205 1.9452213 1.98788317
 2.03501492 2.09497142 2.15098144 2.2097724 2.27432117 2.33697783
 2.39689737 2.45818412 2.51950998]
```



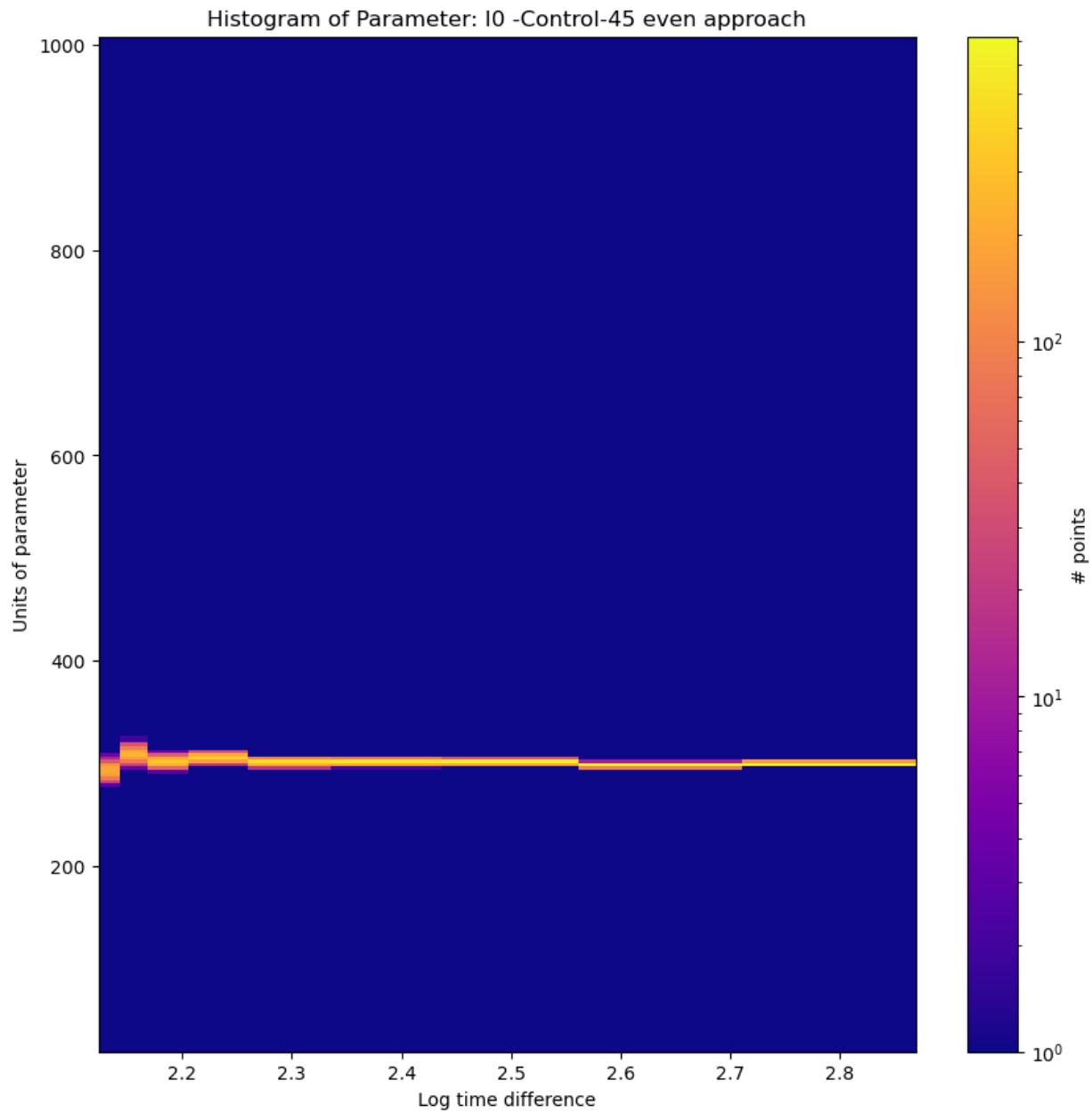
my x edges

```
[0.44639502 0.95154499 1.31774187 1.46911739 1.57081045 1.65856962
 1.72708783 1.80883598 1.8676842 1.92186855 1.98561851 2.04632984
 2.10560611 2.16596571 2.22427618 2.28267561 2.34410219 2.40478666
 2.46419209 2.52344831]
```



my x edges

```
[2.12375816 2.14331966 2.16812462 2.20556295 2.26011657 2.33610728  
2.43650084 2.5618607 2.71012946 2.86903197]
```



```
In [11]: #Plotting histograms for T
aux_list = [list_par_separated_e1[2], list_par_separated_e2[2], list_par_separated_o1[2],
            list_par_separated_o2[2], list_par_separated_c[2]]

list_times = [exp_entropy1.totaltimes(), exp_entropy2.totaltimes(), exp_on_the_fly1.totaltimes(),
              exp_on_the_fly2.totaltimes(), exp_control.totaltimes()]

names_for_approaches = ["Entropy 1 (Selected)", "Entropy 2 (All)", "On-the fly 1", "On-the fly 2",
                        "Control-45 even"]
par_name = "T"

# specify y-edges for all three histograms
y_min, y_max = MyPlots.finding_max_min(aux_list)

for i in range(len(aux_list)):
    MyPlots.plotting_hist_logtime(aux_list[i], names_for_approaches[i], par_name, y_min, y_max)
```

```
#####GMM VERSION COMMENT OUT IF YOU DO NOT HAVE A GMM VERSION

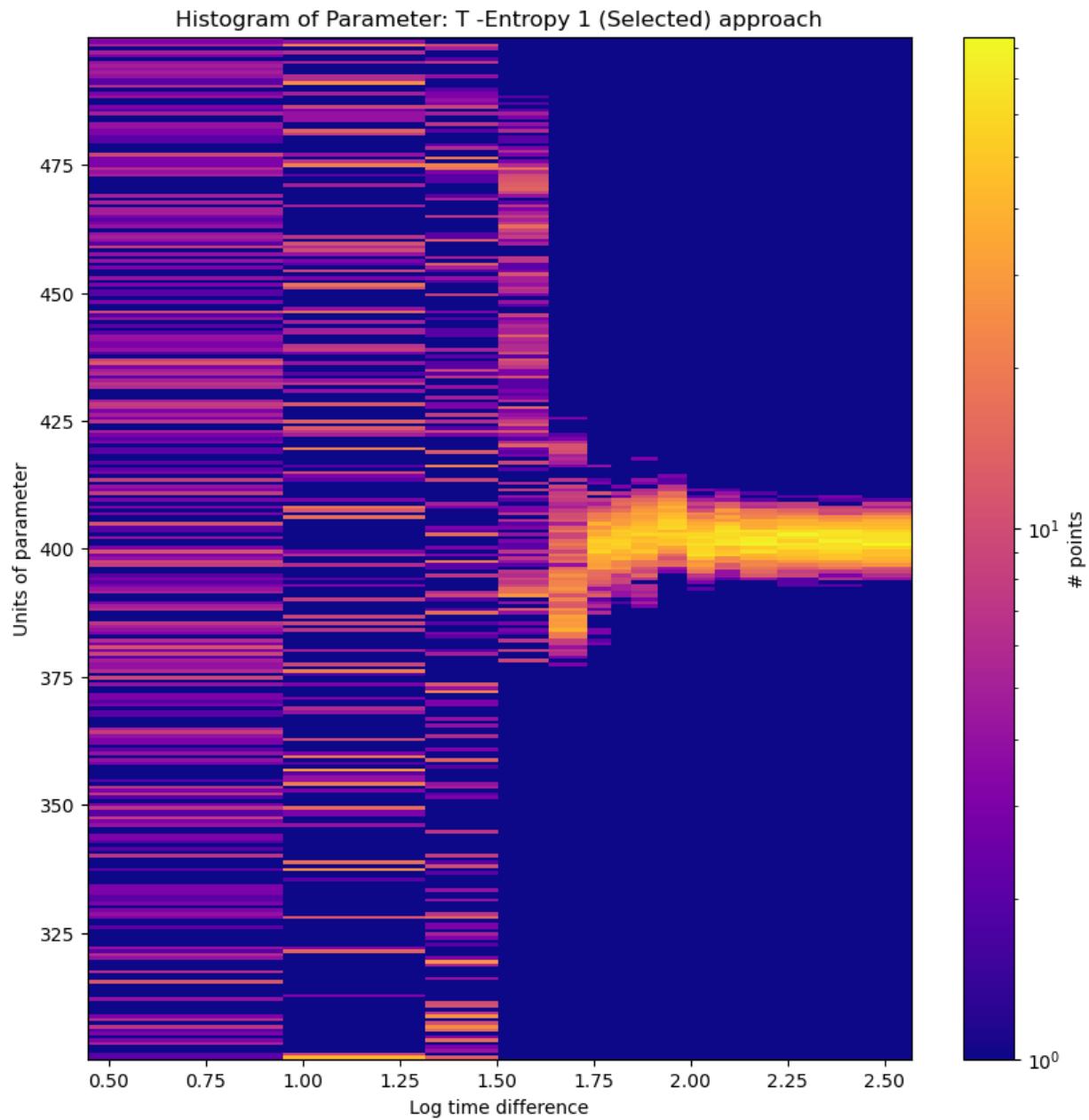
# aux_list = [list_par_separated_e1_gmm[2], list_par_separated_e2_gmm[2], list_par_sep#
#               list_par_separated_o2_gmm[2], list_par_separated_c_gmm[2]]

# names_for_approaches = ["Entropy 1 (Selected) GMM", "Entropy 2 (ALL) GMM", "On-the fl#
#                           "Control-45 even GMM"]
# par_name = "T"

# # specify y-edges for all three histograms
# y_min, y_max = MyPlots.finding_max_min(aux_list)

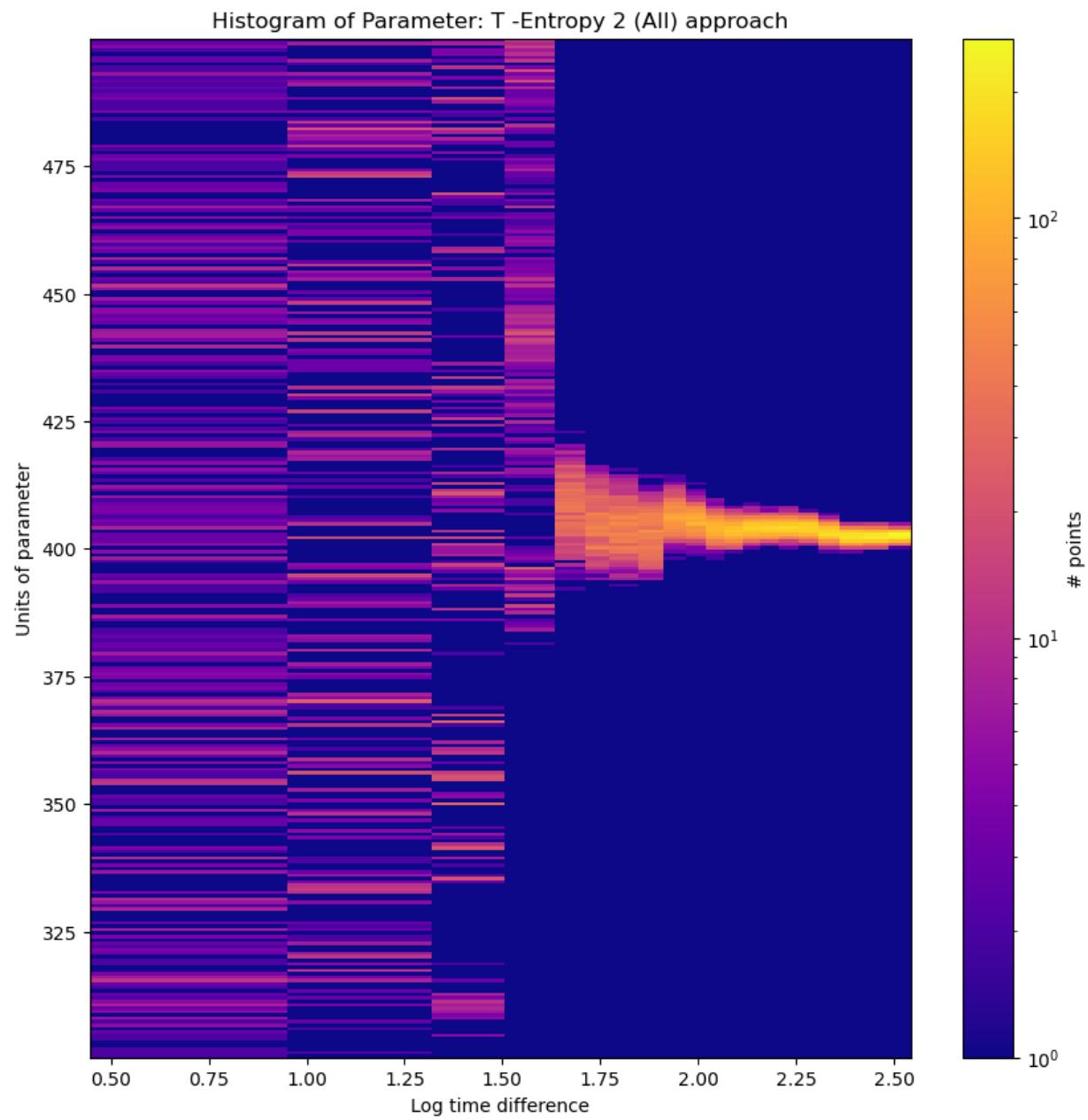
# for i in range(len(aux_list)):
#     MyPlots.plotting_hist(aux_list[i], names_for_approaches[i], par_name, y_min, y_n

my x edges
[0.44639502 0.95154499 1.31774187 1.50557368 1.63498984 1.73417367
 1.7955323 1.84686347 1.91483462 1.98918063 2.06119796 2.12836379
 2.22187384 2.32862824 2.44108604 2.56856595]
```



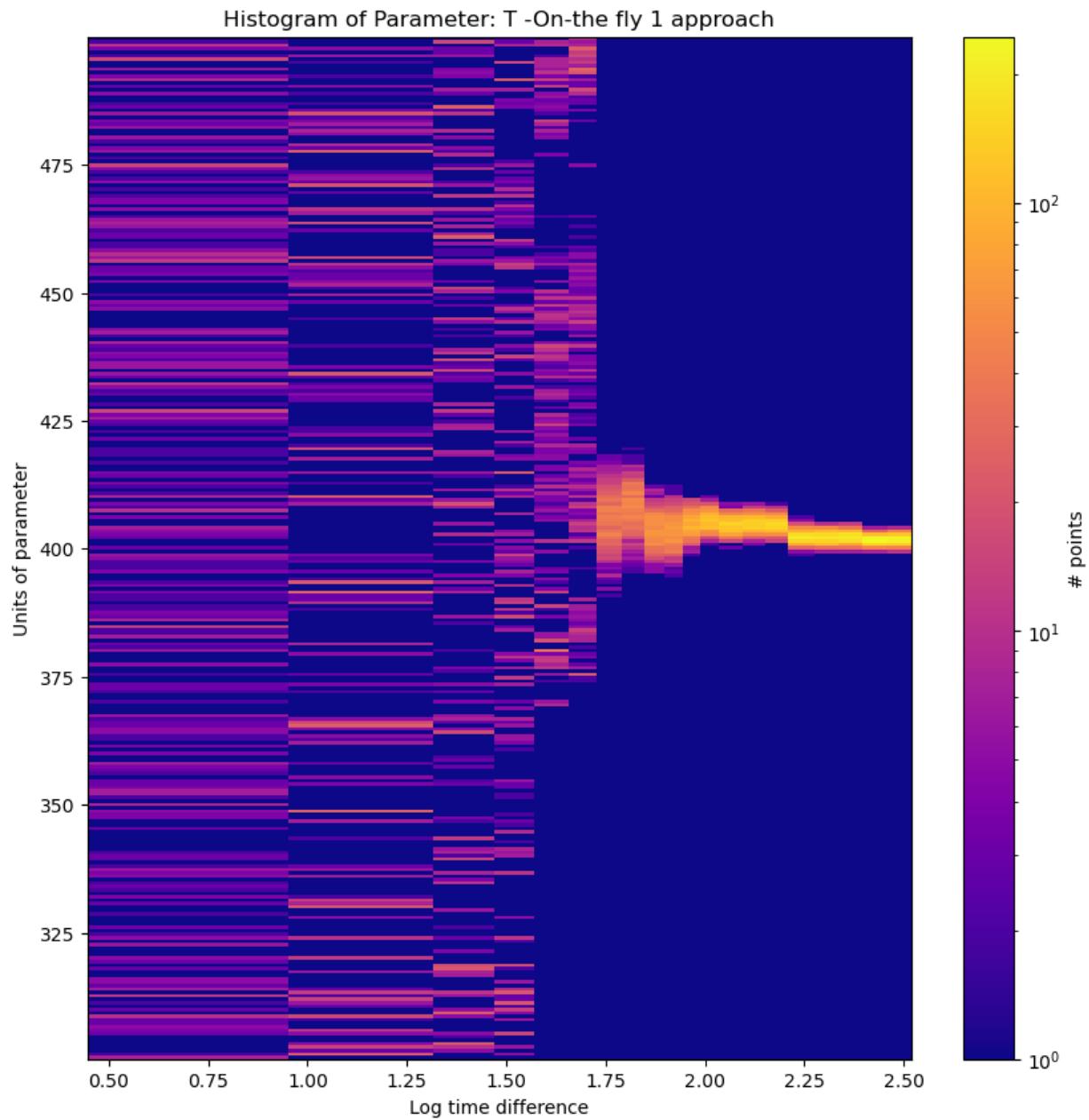
my x edges

```
[0.44639502 0.95154499 1.31774187 1.50557368 1.63498984 1.71129492
 1.77265356 1.84686347 1.91016642 1.96791968 2.01978708 2.06722129
 2.11291436 2.15680688 2.2041022 2.25523444 2.30786839 2.36212059
 2.4234927 2.48599068 2.54348622]
```



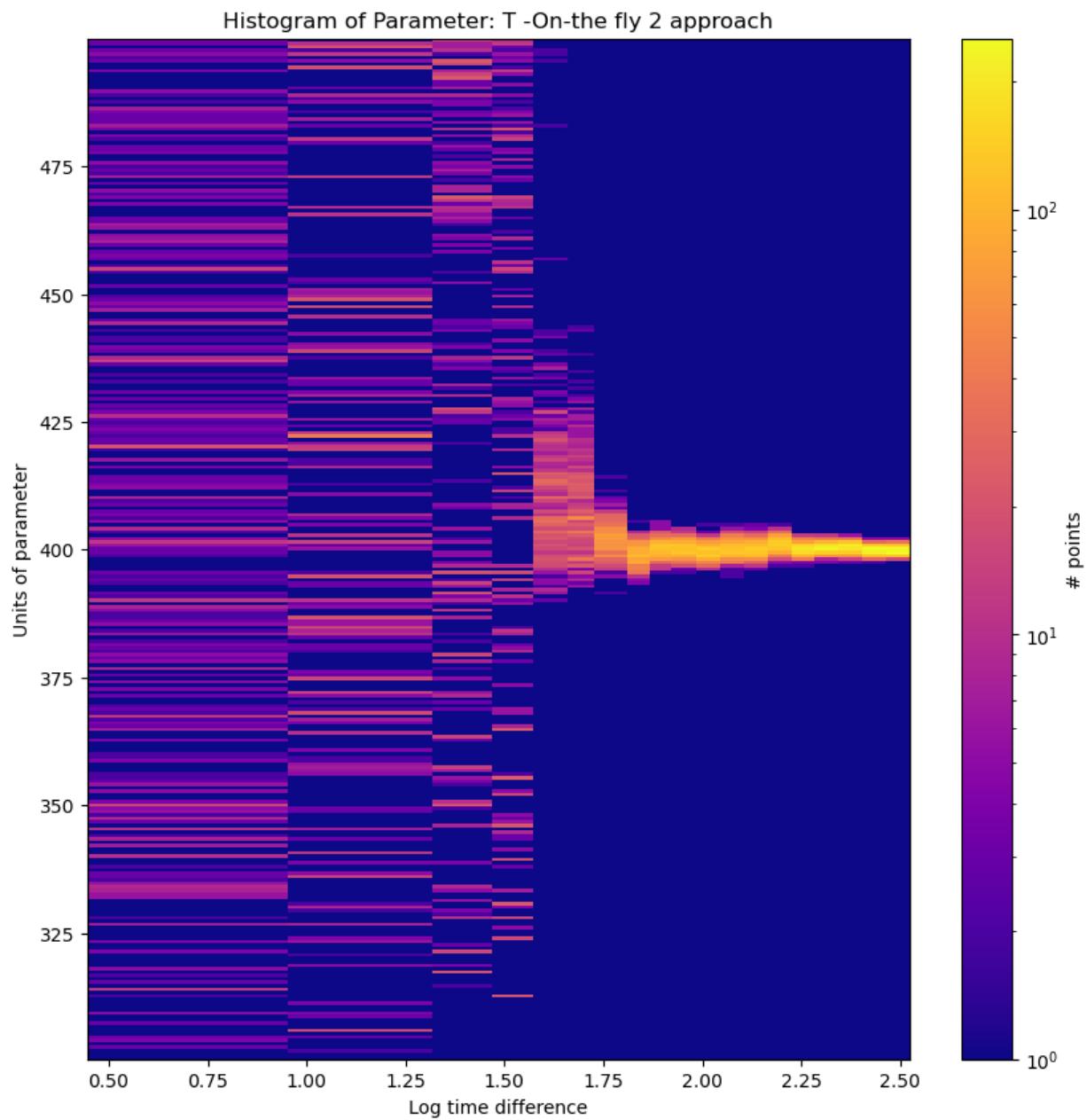
my_x_edges

```
[0.44639502 0.95154499 1.31774187 1.46825687 1.56930922 1.65735485
 1.72604662 1.79184115 1.84678641 1.89944205 1.9452213 1.98788317
 2.03501492 2.09497142 2.15098144 2.2097724 2.27432117 2.33697783
 2.39689737 2.45818412 2.51950998]
```



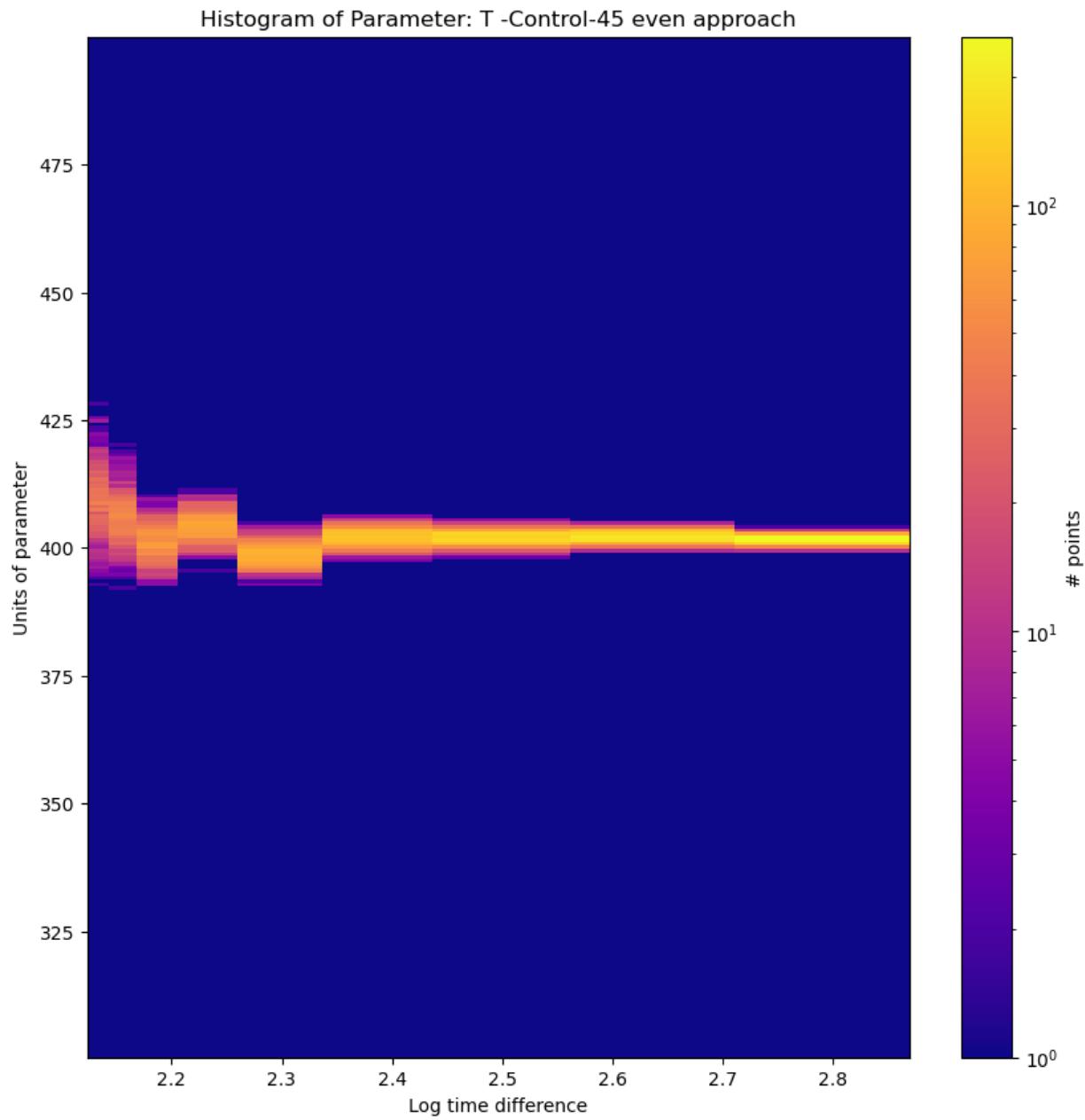
```
my x edges
```

```
[0.44639502 0.95154499 1.31774187 1.46911739 1.57081045 1.65856962
 1.72708783 1.80883598 1.8676842 1.92186855 1.98561851 2.04632984
 2.10560611 2.16596571 2.22427618 2.28267561 2.34410219 2.40478666
 2.46419209 2.52344831]
```



```
my_x_edges
```

```
[2.12375816 2.14331966 2.16812462 2.20556295 2.26011657 2.33610728
 2.43650084 2.5618607 2.71012946 2.86903197]
```



In []:

```
#Plotting histograms for Phi0
aux_list = [list_par_separated_e1[3], list_par_separated_e2[3], list_par_separated_o1[3],
            list_par_separated_o2[3], list_par_separated_c[3]]

list_times = [exp_entropy1.totaltimes(), exp_entropy2.totaltimes(), exp_on_the_fly1.totaltimes(),
              exp_on_the_fly2.totaltimes(), exp_control.totaltimes()]

names_for_approaches = ["Entropy 1 (Selected)", "Entropy 2 (All)", "On-the fly 1", "On-the fly 2", "Control-45 even"]
par_name = "Phi0"

# specify y-edges for all three histograms
y_min, y_max = MyPlots.finding_max_min(aux_list)
```

```
for i in range(len(aux_list)):
    MyPlots.plotting_hist_logtime(aux_list[i], names_for_approaches[i], par_name, y_mi

#####
#####GMM VERSION COMMENT OUT IF YOU DO NOT HAVE A GMM VERSION

# aux_list = [list_par_separated_e1_gmm[3], list_par_separated_e2_gmm[3], list_par_sep
#             list_par_separated_o2_gmm[3], list_par_separated_c_gmm[3]]

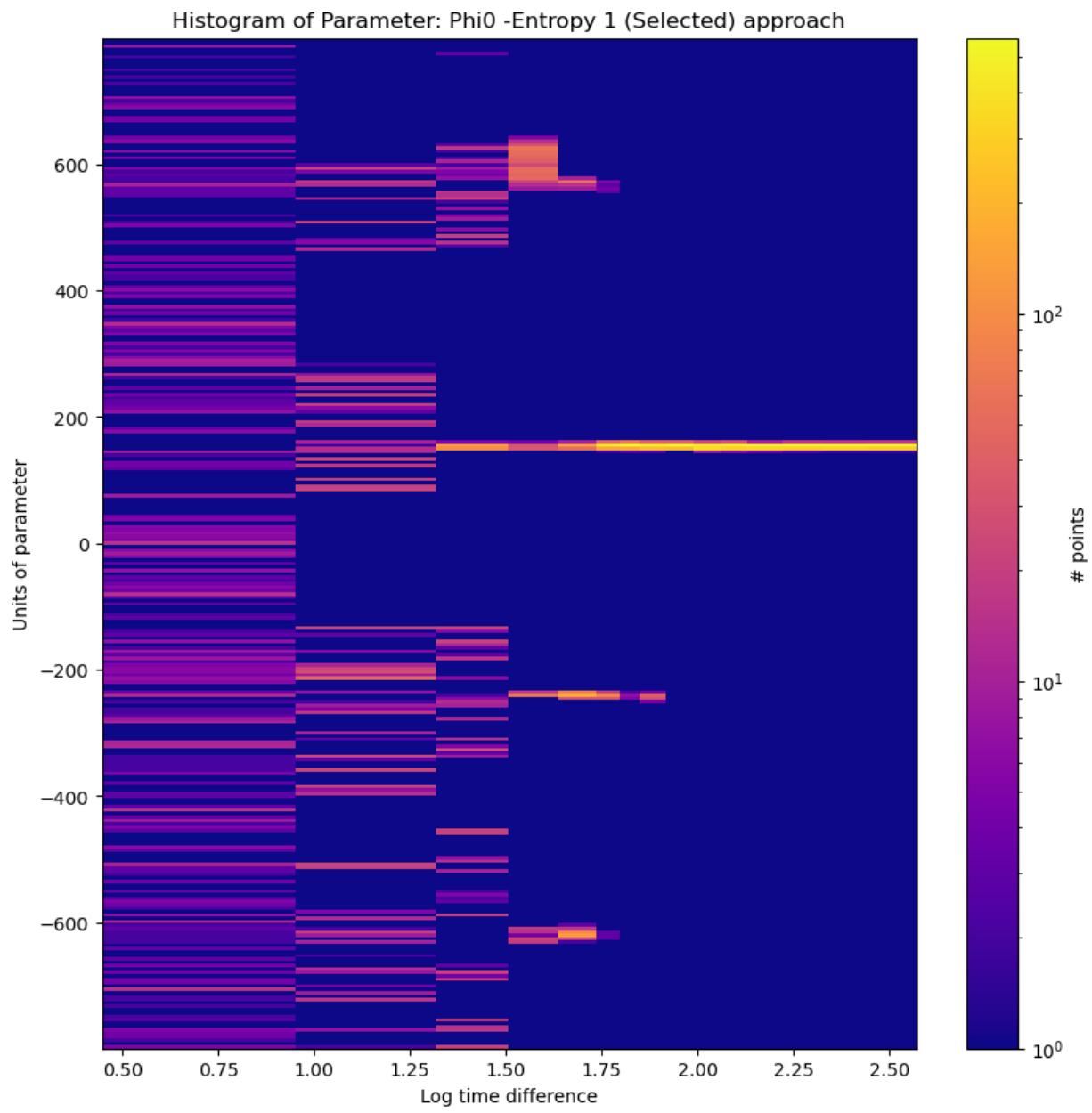
# names_for_approaches = ["Entropy 1 (Selected) GMM", "Entropy 2 (ALL) GMM", "On-the fl
#                         "Control-45 even GMM"]
# par_name = "Phi0"

#
# # specify y-edges for all three histograms
# y_min, y_max = MyPlots.finding_max_min(aux_list)

# for i in range(len(aux_list)):
#     MyPlots.plotting_hist(aux_list[i], names_for_approaches[i], par_name, y_min, y_n
```

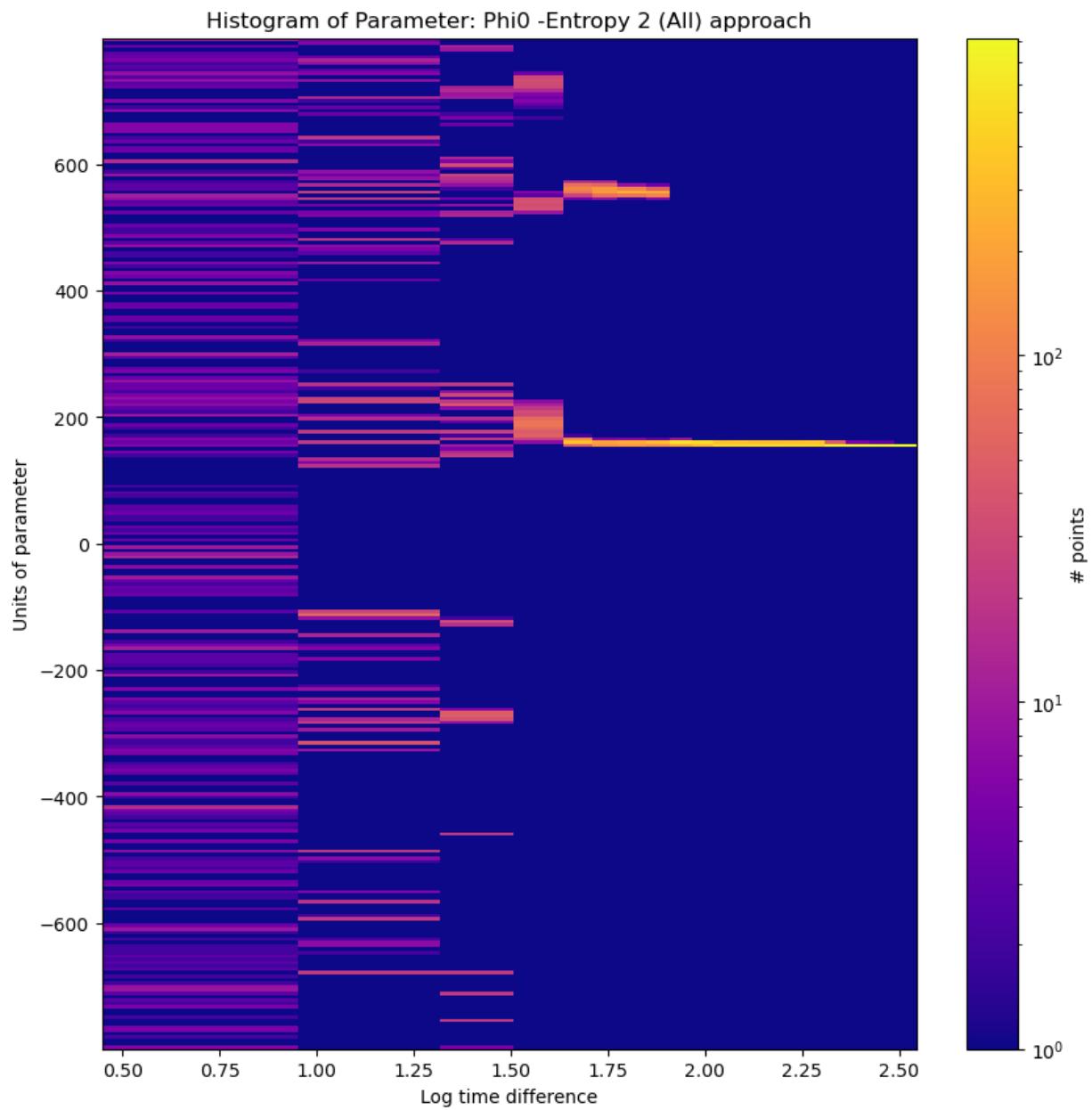
my x edges

```
[0.44639502 0.95154499 1.31774187 1.50557368 1.63498984 1.73417367
 1.7955323 1.84686347 1.91483462 1.98918063 2.06119796 2.12836379
 2.22187384 2.32862824 2.44108604 2.56856595]
```



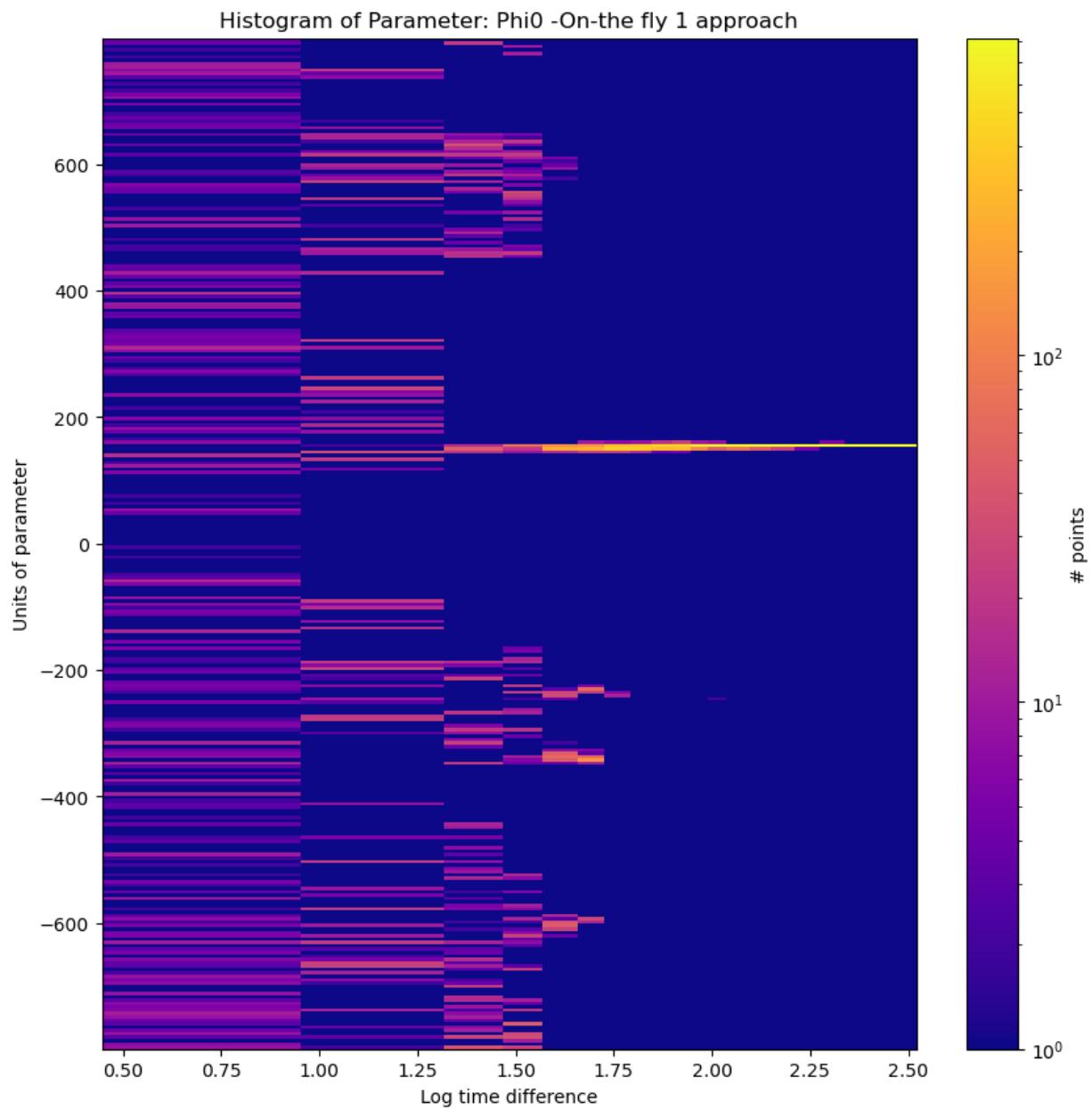
my x edges

```
[0.44639502 0.95154499 1.31774187 1.50557368 1.63498984 1.71129492
 1.77265356 1.84686347 1.91016642 1.96791968 2.01978708 2.06722129
 2.11291436 2.15680688 2.2041022 2.25523444 2.30786839 2.36212059
 2.4234927 2.48599068 2.54348622]
```



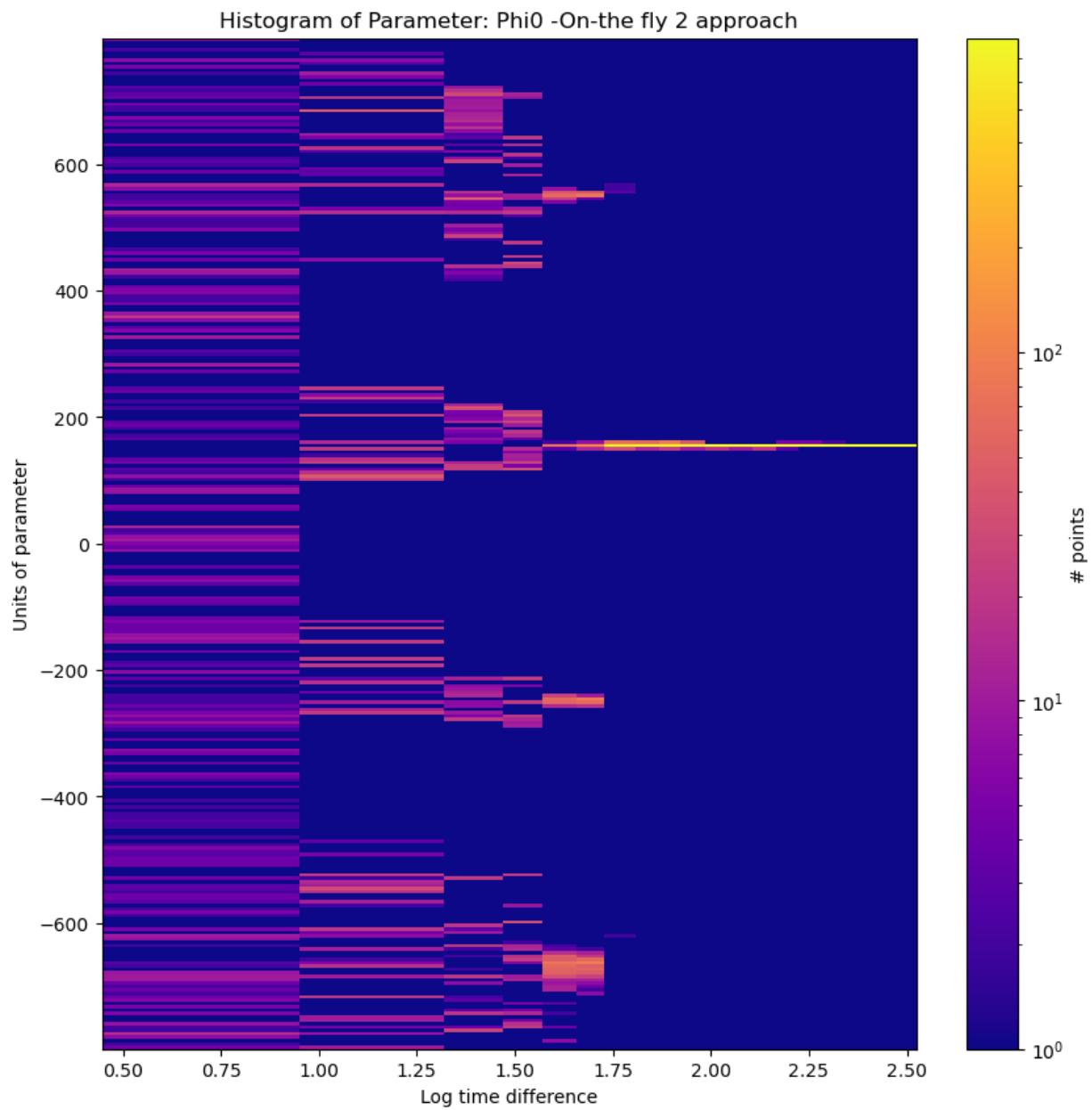
my x edges

```
[0.44639502 0.95154499 1.31774187 1.46825687 1.56930922 1.65735485
 1.72604662 1.79184115 1.84678641 1.89944205 1.9452213 1.98788317
 2.03501492 2.09497142 2.15098144 2.2097724 2.27432117 2.33697783
 2.39689737 2.45818412 2.51950998]
```



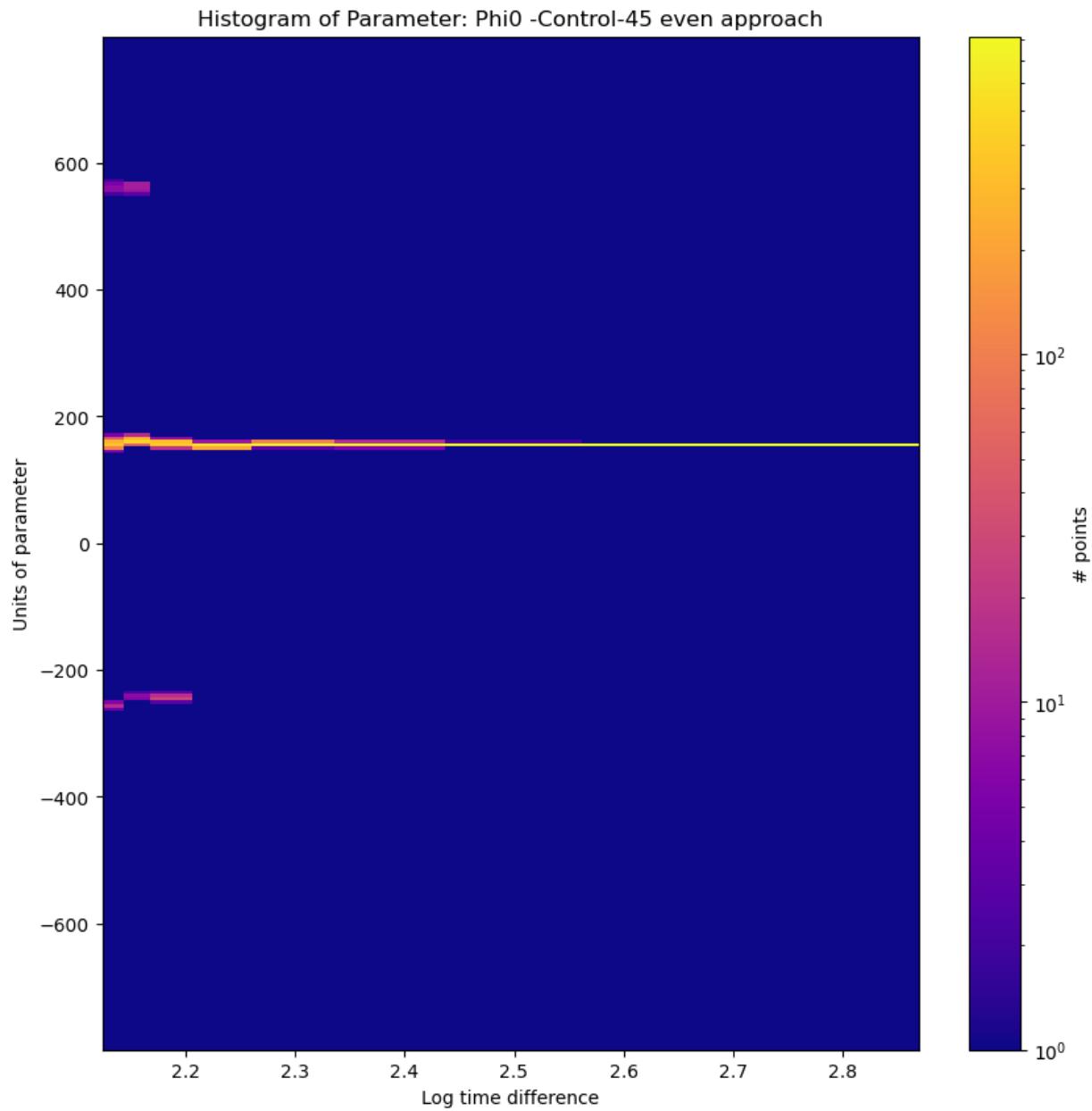
my x edges

```
[0.44639502 0.95154499 1.31774187 1.46911739 1.57081045 1.65856962
 1.72708783 1.80883598 1.8676842 1.92186855 1.98561851 2.04632984
 2.10560611 2.16596571 2.22427618 2.28267561 2.34410219 2.40478666
 2.46419209 2.52344831]
```



my x edges

```
[2.12375816 2.14331966 2.16812462 2.20556295 2.26011657 2.33610728  
2.43650084 2.5618607 2.71012946 2.86903197]
```



10.3 Entropy V.S Time

```
In [13]: #Notice we could also print the total entropy, instead, of the marginalized entropy.
#On-the-fly always deals with the total entropy. You cannot choose a parameter of interest
times = [exp_entropy1.totaltimes(), exp_on_the_fly1.totaltimes(),
         exp_on_the_fly2.totaltimes(), exp_control.totaltimes(), exp_entropy2.totaltime()]

entropies = [exp_entropy1.entropy(), exp_on_the_fly1.entropy(),
            exp_on_the_fly2.entropy(), exp_control.entropy(), exp_entropy2.entropy()]

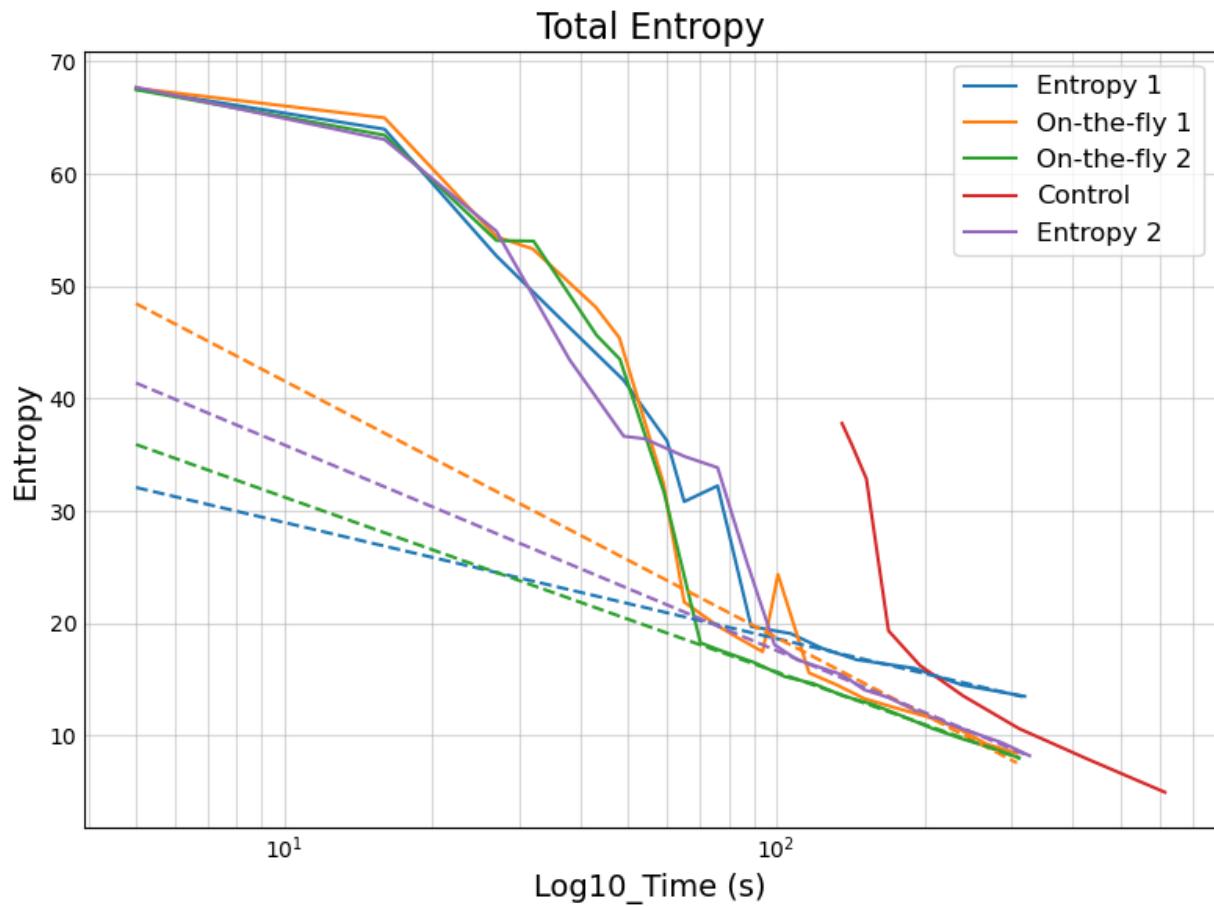
list_names = ["Entropy 1 ", "On-the-fly 1", "On-the-fly 2", "Control", "Entropy 2"]

MyPlots.plot_entropy_times(times, entropies, list_names, "Total Entropy")

#C1 blue entrop
#C2 yellow on the
#C3 green on the 2
```

```
#C4 red control
#C5 Purple entropy2
```

```
#Total Entropy
```



```
In [14]: #DELETE LATER THIS WORK ONLY IF YOU HAVE A GMM
```

```
#Entropy 1 Comparisson
#Blue fast mvn
#Orange gmm
```

```
# MyPlots.plot_entropy_times([exp_entropy1.totaltimes(), exp_entropy1_gmm.totaltimes()]
#                               [exp_entropy1.entropy(), exp_entropy1_gmm.entropy()])
# plt.savefig("entropy1 total entropy mvn_blue gmm_orange.png")
```

```
In [15]: #DELETE LATER THIS WORK ONLY IF YOU HAVE A GMM
```

```
#Entropy 2 Comparisson
#Blue fast mvn
#Orange gmm
```

```
#COMMENT OUT IF YOU DO NOT HAVE A GMM VERSION
```

```
# MyPlots.plot_entropy_times([exp_entropy2.totaltimes(), exp_entropy2_gmm.totaltimes()]
#                               [exp_entropy2.entropy(), exp_entropy2_gmm.entropy()])
```

```
In [16]: #On the fly Comparisson
#Blue fast mvn
#Orange gmm
```

```
# MyPlots.plot_entropy_times([exp_on_the_fly1.totaltimes(), exp_on_the_fly1_gmm.totaltimes(),
#                             [exp_on_the_fly1.entropy(), exp_on_the_fly1_gmm.entropy()])
```

In [17]: #On the fly 2- Total Entropy Comparisson
#Blue fast mvn
#Orange gmm

```
# MyPlots.plot_entropy_times([exp_on_the_fly2.totaltimes(), exp_on_the_fly2_gmm.totaltimes(),
#                             [exp_on_the_fly2.entropy(), exp_on_the_fly2_gmm.entropy()])
```

In []:

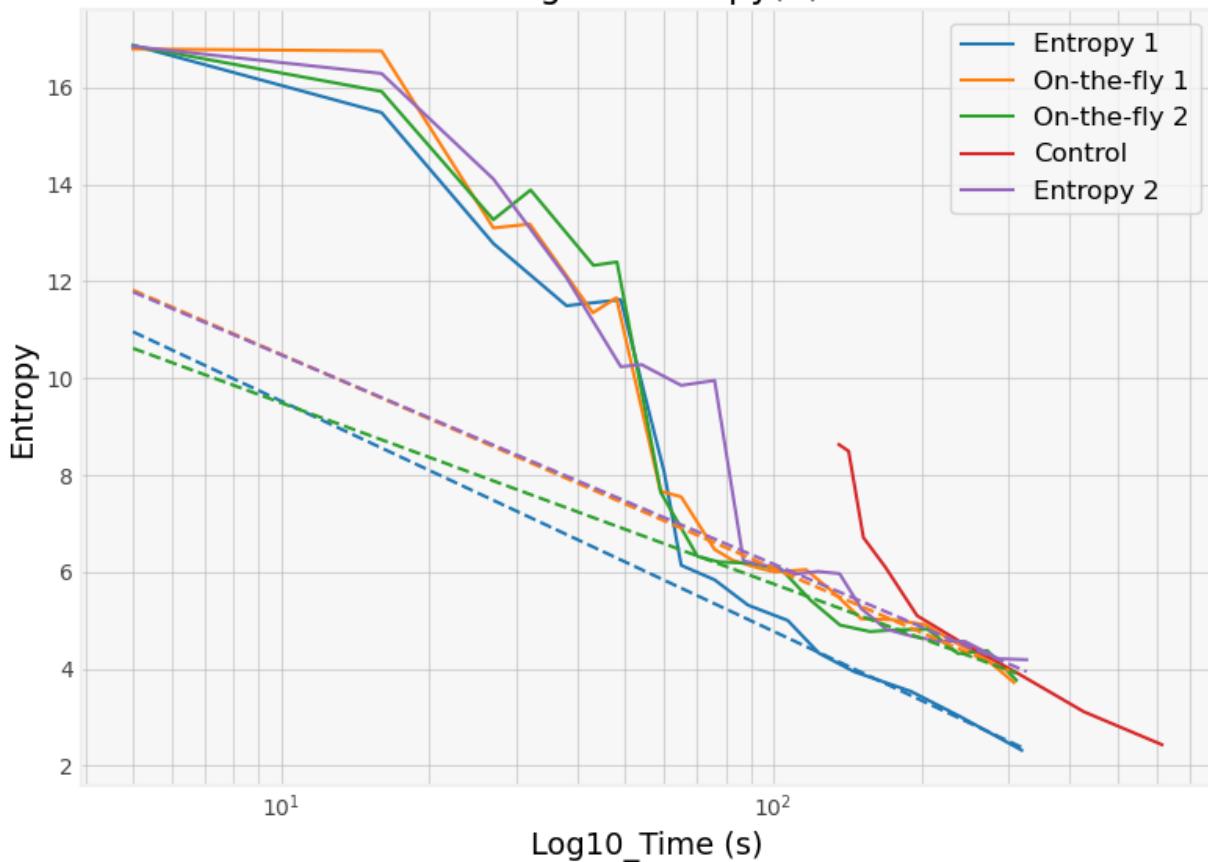
In [18]: #Ask REMEMBER THE RETURN OF ENOTROPY_MARG() BC entropy2 does not have a marginal
times = [exp_entropy1.totaltimes(), exp_on_the_fly1.totaltimes(),
 exp_on_the_fly2.totaltimes(), exp_control.totaltimes(), exp_entropy2.totaltimes()]

list_names = ["Entropy 1 ", "On-the-fly 1", "On-the-fly 2", "Control", "Entropy 2"]

#Rememeber that the marginal entropy for entropy2 apporach is the same as total entropy
#is not selecting any parameters. Thus, we calculate here using the select of any of the
#approaches that it must be same.
entro = [calc_entropy(this_pts, select_pars=exp_on_the_fly2.settings.sel,
 options=exp_entropy2.settings.entropy_options)[0] for this_pts in exp_
 _times]

entropies = [exp_entropy1.entropy_marg(), exp_on_the_fly1.entropy_marg(),
 exp_on_the_fly2.entropy_marg(), exp_control.entropy_marg(), entro]
MyPlots.plot_entropy_times(times, entropies, list_names, "Marginal Entropy(A)")

Marginal Entropy(A)



```
In [19]: #DELETE LATER THIS WORK ONLY IF YOU HAVE A GMM
```

```
#Entropy 1 Marginal Comparisson
#Blue fast mvn
#Orange gmm
```

```
# MyPlots.plot_entropy_times([exp_entropy1.totaltimes(), exp_entropy1_gmm.totaltimes()]
#                           [exp_entropy1.entropy_marg(), exp_entropy1_gmm.entropy_marg()])
```

```
In [20]: #Entropy 2- Marginal Comparisson
```

```
# #####Comment out unless you have a GMM version
# gmm_setting= {'method': 'gmm', 'n_components': None}
# entro1 = recalculating_entropy(exp_entropy2, exp_on_the_fly2.settings.sel,gmm_setting)
# entro2 = recalculating_entropy(exp_entropy2_gmm, exp_on_the_fly2.settings.sel,gmm_setting)
# #Blue fast mvn
# #Orange gmm
# MyPlots.plot_entropy_times([exp_entropy2.totaltimes(), exp_entropy2_gmm.totaltimes()]
#                           [entro1, entro2])
```

```
In [21]: #DELETE LATER THIS WORK ONLY IF YOU HAVE A GMM
```

```
#On the fly 1 Marginal Comparisson
#Blue fast mvn
#Orange gmm

#Comment out if you do not have GMM version
```

```
# MyPlots.plot_entropy_times([exp_on_the_fly1.totaltimes(), exp_on_the_fly1_gmm.totalt
#                                         [exp_on_the_fly1.entropy_marg(), exp_on_the_fly1_gmm.entr
```

In [22]: *#DELETE LATER THIS WORK ONLY IF YOU HAVE A GMM*

```
#On the fly 1 Marginal Comparisson
#Blue fast mvn
#Orange gmm
```

```
# MyPlots.plot_entropy_times([exp_on_the_fly2.totaltimes(), exp_on_the_fly2_gmm.totalt
#                                         [exp_on_the_fly2.entropy_marg(), exp_on_the_fly2_gmm.entr
```

Helper Functions for Estimator of PDF and Log-likelihoods

These function will be used in the remaining sections

.....

estimator_avg_at

Note: In my opinion, the estimator_avg_at causes an error in some cases bc at some points of the iteration the values are concentrate around the ground truth. Thus, there are no many points to take on the left or right of the interval. So the checking below alert us!

```
if (left_value_index < 0) or (right_value_index > (NUMBER_SAMPLES - 1)): Alert!
```

"" Note: Notice compute_lokelihoods assumes independence of the parameters (columns). It is using David's estimator for pdf. Ask But can we assume that?

""" Notes likelihood_helper_kde:

1. Using KDE to estimate the pdf. Notice that KDE is a non-parametric estimator. However, it assumes independent samples; this could be a problem. Remember each sample is measurement point in the main loop is chosen based in the previous results.
2. Ask about the bandwidth for this function. In the next functions, we use a kde in a single feature (a parameter or a y at a given measurement place); so we can use the std as the bandwidth. We use the std bc all the other methods s.a. silverman or scott gave us terrible estimators. Thus, it may be the case that they are terrible estimators since we are using silverman. In this case, I am using "silverman". Since it is a multidimensional pdf I cannot use just the std. Should I tried to use something similar to the std?
""" #likelihood_helper_kde

10.4 Estimator of Log Likelihoods for the Y's

We will use David's estimator and a multidimensional Kde

Using David's Estimator

In [23]:

```
#all_total_Likelihoods
#At each iteration, we have a y-profiles. We compute the likelihood at each x and add
Sum_likelihoods_at_each_iter_entropy1_for_ys = MyPlots.compute_likelihoods(exp_entropy1,
                                                               exp_entropy2)
Sum_likelihoods_at_each_iter_entropy2_for_ys = MyPlots.compute_likelihoods(exp_entropy1,
                                                               exp_entropy2)
Sum_likelihoods_at_each_iter_on_the_fly1_for_ys = MyPlots.compute_likelihoods(exp_on_the_fly1,
                                                               exp_on_the_fly2)
Sum_likelihoods_at_each_iter_on_the_fly2_for_ys = MyPlots.compute_likelihoods(exp_on_the_fly1,
                                                               exp_on_the_fly2)

Sum_likelihoods_at_each_iter_control_for_ys = MyPlots.compute_likelihoods(exp_control1,
                                                               exp_control2)
#Sum_likelihoods_at_each_iter_control_for_ys = compute_likelihoods(total_yprofs_control)
```

```
ERROR: the ground truth to estimate is too close to one extreme. So, we do not have e
nought samples eitheron the left or right of the ground truth
This is the left index
839
this is righth index
840
ERROR: the ground truth to estimate is too close to one extreme. So, we do not have e
nought samples eitheron the left or right of the ground truth
This is the left index
839
this is righth index
840
ERROR: the ground truth to estimate is too close to one extreme. So, we do not have e
nought samples eitheron the left or right of the ground truth
This is the left index
839
this is righth index
840
ERROR: the ground truth to estimate is too close to one extreme. So, we do not have e
nought samples eitheron the left or right of the ground truth
This is the left index
839
this is righth index
840
ERROR: the ground truth to estimate is too close to one extreme. So, we do not have e
nought samples eitheron the left or right of the ground truth
This is the left index
839
this is righth index
840
ERROR: the ground truth to estimate is too close to one extreme. So, we do not have e
nought samples eitheron the left or right of the ground truth
This is the left index
839
this is righth index
840
ERROR: the ground truth to estimate is too close to one extreme. So, we do not have e
nought samples eitheron the left or right of the ground truth
This is the left index
839
this is righth index
840
ERROR: the ground truth to estimate is too close to one extreme. So, we do not have e
nought samples eitheron the left or right of the ground truth
This is the left index
839
this is righth index
840
C:\Users\rober\anaconda3\lib\site-packages\numpy\core\fromnumeric.py:3432: RuntimeWar
ning: Mean of empty slice.
    return _methods._mean(a, axis=axis, dtype=dtype,
C:\Users\rober\anaconda3\lib\site-packages\numpy\core\_methods.py:190: RuntimeWarnin
g: invalid value encountered in double_scalars
    ret = ret.dtype.type(ret / rcount)
```

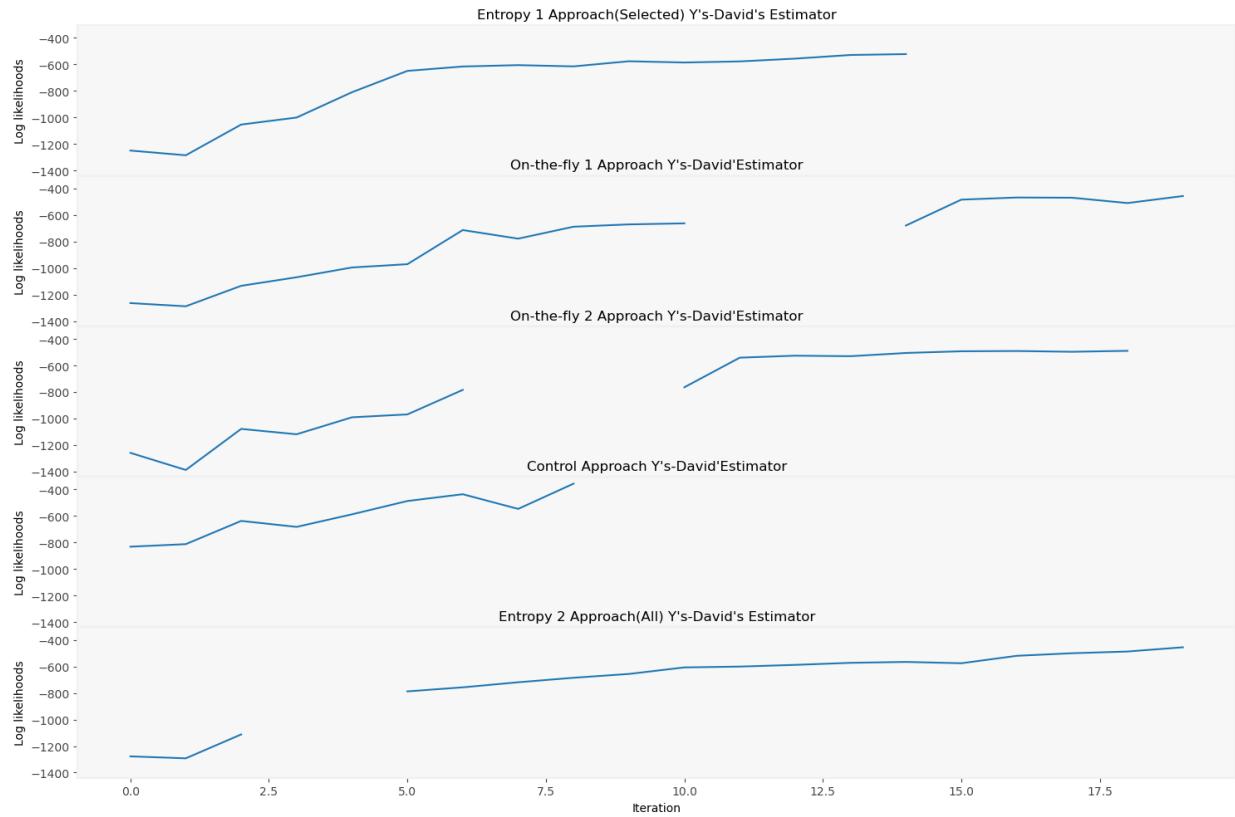


```
ERROR: the ground truth to estimate is too close to one extreme. So, we do not have e
nought samples eitheron the left or right of the ground truth
This is the left index
797
this is righth index
798
ERROR: the ground truth to estimate is too close to one extreme. So, we do not have e
nought samples eitheron the left or right of the ground truth
This is the left index
797
this is righth index
798
ERROR: the ground truth to estimate is too close to one extreme. So, we do not have e
nought samples eitheron the left or right of the ground truth
This is the left index
797
this is righth index
798
ERROR: the ground truth to estimate is too close to one extreme. So, we do not have e
nought samples eitheron the left or right of the ground truth
This is the left index
797
this is righth index
798
ERROR: the ground truth to estimate is too close to one extreme. So, we do not have e
nought samples eitheron the left or right of the ground truth
This is the left index
797
this is righth index
798
ERROR: the ground truth to estimate is too close to one extreme. So, we do not have e
nought samples eitheron the left or right of the ground truth
This is the left index
797
this is righth index
798
```

```
In [24]: MyPlots.plot_likelihood_iterations(5,1, [Sum_likelihoods_at_each_iter_entropy1_for_ys,
                                                Sum_likelihoods_at_each_iter_on_the_fly1_for_
                                                Sum_likelihoods_at_each_iter_on_the_fly2_for_
                                                Sum_likelihoods_at_each_iter_control_for_ys,
                                                Sum_likelihoods_at_each_iter_entropy2_for_ys
                                               ,["Entropy 1 Approach(Selected) Y's-David's Estimator",
                                                 "On-the-fly 1 Approach Y's-David'Estimator",
                                                 "On-the-fly 2 Approach Y's-David'Estimator",
                                                 "Control Approach Y's-David'Estimator",
                                                 "Entropy 2 Approach(All) Y's-David's Estimator"]])
```

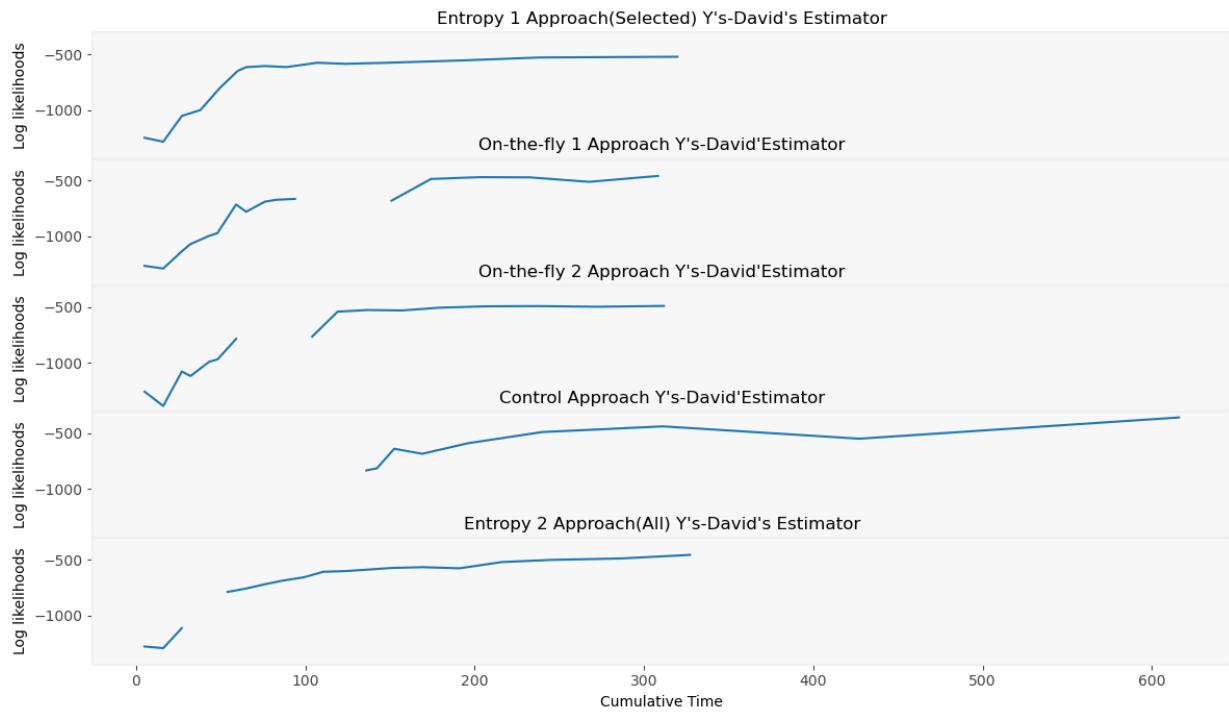
#REMEMBER: CONTROL TAKES A DIFFERENT NUMBER OF POINTS THAN THE OTHER APPORACHES.

Run Plots



```
In [25]: #Add Later the other apporaches to this plot
#[exp_entropy.totaltimes()[1:], exp_on_the_fly.totaltimes()[1:], exp_control.totaltime
MyPlots.plot_likelihood_time(5,1, [Sum_likelihoods_at_each_iter_entropy1_for_ys,
                                  Sum_likelihoods_at_each_iter_on_the_fly1_for_
                                  Sum_likelihoods_at_each_iter_on_the_fly2_for_
                                  Sum_likelihoods_at_each_iter_control_for_ys,
                                  Sum_likelihoods_at_each_iter_entropy2_for_ys],
                               [exp_entropy1.totaltimes(), exp_on_the_fly1.totaltimes(),
                                exp_on_the_fly2.totaltimes(), exp_control.totaltimes(),
                                exp_entropy2.totaltimes()],
                               ["Entropy 1 Approach(Selected) Y's-David's Estimator",
                                "On-the-fly 1 Approach Y's-David'Estimator",
                                "On-the-fly 2 Approach Y's-David'Estimator",
                                "Control Approach Y's-David'Estimator",
                                "Entropy 2 Approach(All) Y's-David's Estimator"])
```

Run Plots



Using Multi-Dimensional KDE

```
In [26]: #all_total_likelihoods
Sum_likelihoods_at_each_iter_entropy1_for_ys_kde = MyPlots.likelihood_helper_kde(exp_en
exp_er

Sum_likelihoods_at_each_iter_entropy2_for_ys_kde = MyPlots.likelihood_helper_kde(exp_en
exp_er

Sum_likelihoods_at_each_iter_on_the_fly1_for_ys_kde = MyPlots.likelihood_helper_kde(exp_en
exp_er

Sum_likelihoods_at_each_iter_on_the_fly2_for_ys_kde = MyPlots.likelihood_helper_kde(exp_en
exp_er

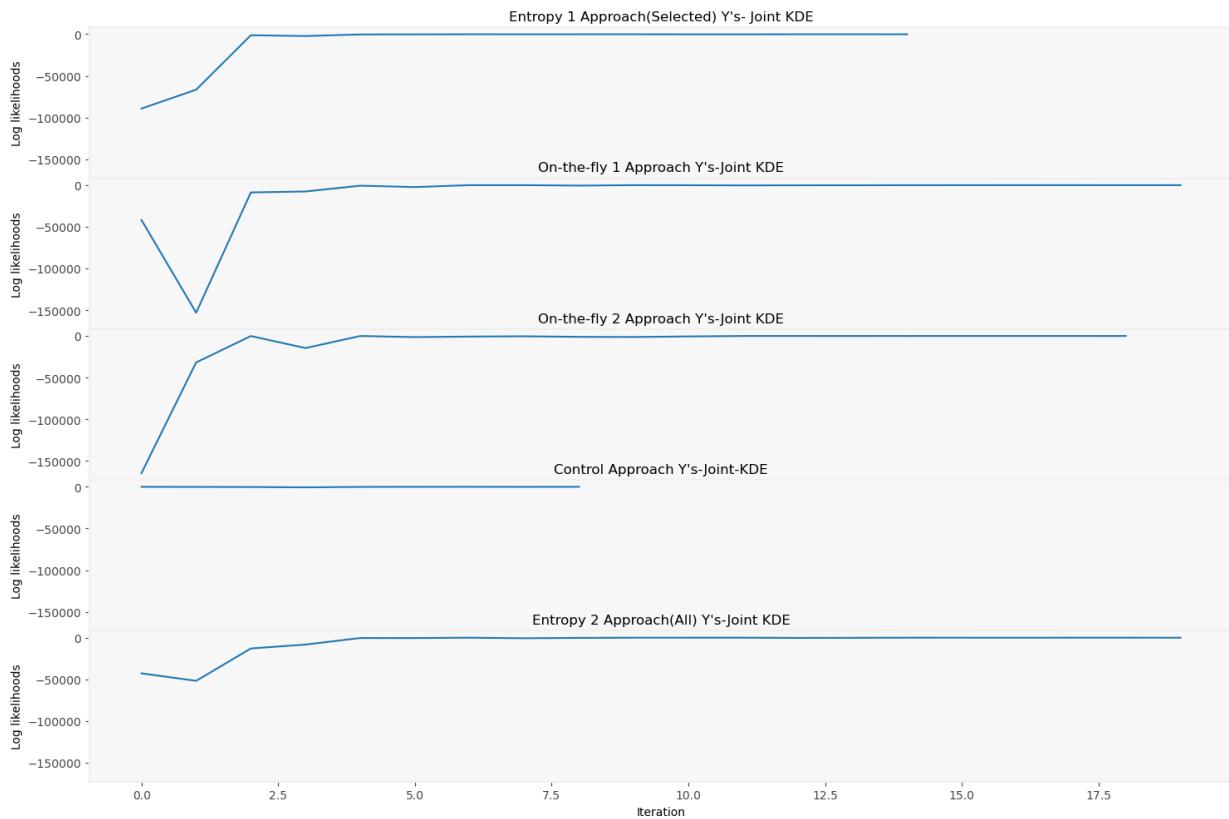
Sum_likelihoods_at_each_iter_control_for_ys_kde = MyPlots.likelihood_helper_kde(exp_co
exp_cr

#Sum_likelihoods_at_each_iter_control_for_ys_kde = likelihood_helper_kde(total_yprof_
```

```
C:\Users\rober\FINAL NIST PROJECT\datastruct.py:346: VisibleDeprecationWarning: Creating an ndarray from ragged nested sequences (which is a list-or-tuple of lists-or-tuples-or ndarrays with different lengths or shapes) is deprecated. If you meant to do this, you must specify 'dtype=object' when creating the ndarray.
    return np.array(all_yprof)
C:\Users\rober\FINAL NIST PROJECT\datastruct.py:346: VisibleDeprecationWarning: Creating an ndarray from ragged nested sequences (which is a list-or-tuple of lists-or-tuples-or ndarrays with different lengths or shapes) is deprecated. If you meant to do this, you must specify 'dtype=object' when creating the ndarray.
    return np.array(all_yprof)
C:\Users\rober\FINAL NIST PROJECT\datastruct.py:346: VisibleDeprecationWarning: Creating an ndarray from ragged nested sequences (which is a list-or-tuple of lists-or-tuples-or ndarrays with different lengths or shapes) is deprecated. If you meant to do this, you must specify 'dtype=object' when creating the ndarray.
    return np.array(all_yprof)
C:\Users\rober\FINAL NIST PROJECT\datastruct.py:346: VisibleDeprecationWarning: Creating an ndarray from ragged nested sequences (which is a list-or-tuple of lists-or-tuples-or ndarrays with different lengths or shapes) is deprecated. If you meant to do this, you must specify 'dtype=object' when creating the ndarray.
    return np.array(all_yprof)
```

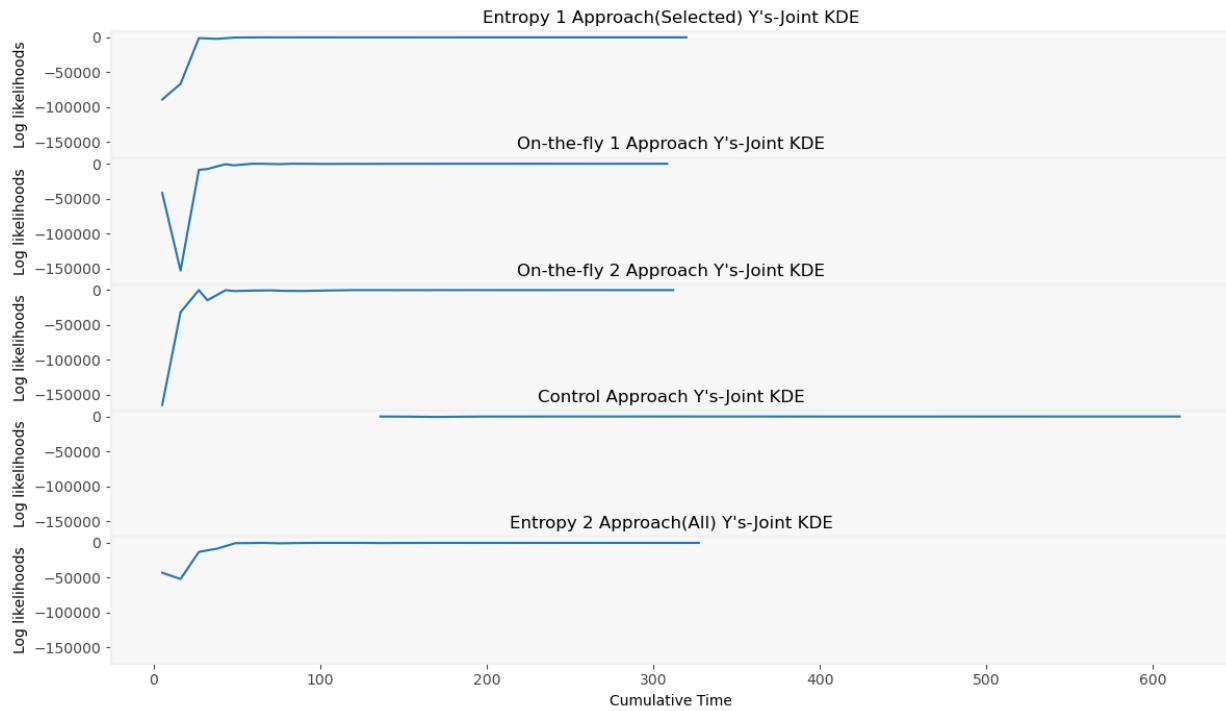
In [27]: *#Add Later the other approaches to this plot*

```
MyPlots.plot_likelihood_iterations(5,1
    ,[Sum_likelihoods_at_each_iter_entropy1_for_ys_kde,
     Sum_likelihoods_at_each_iter_on_the_fly1_for_ys_kde,
     Sum_likelihoods_at_each_iter_on_the_fly2_for_ys_kde,
     Sum_likelihoods_at_each_iter_control_for_ys_kde,
     Sum_likelihoods_at_each_iter_entropy2_for_ys_kde]
    , ["Entropy 1 Approach(Selected) Y's- Joint KDE",
       "On-the-fly 1 Approach Y's-Joint KDE",
       "On-the-fly 2 Approach Y's-Joint KDE",
       "Control Approach Y's-Joint-KDE",
       "Entropy 2 Approach(All) Y's-Joint KDE"])
```



```
In [28]: MyPlots.plot_likelihood_time(5,1
    , [Sum_likelihoods_at_each_iter_entropy1_for_ys_kde,
     Sum_likelihoods_at_each_iter_on_the_fly1_for_ys_kde,
     Sum_likelihoods_at_each_iter_on_the_fly2_for_ys_kde,
     Sum_likelihoods_at_each_iter_control_for_ys_kde,
     Sum_likelihoods_at_each_iter_entropy2_for_ys_kde
    ],
    [ exp_entropy1.totaltimes(),
     exp_on_the_fly1.totaltimes(),
     exp_on_the_fly2.totaltimes(),
     exp_control.totaltimes(),
     exp_entropy2.totaltimes()
    ],
    ["Entropy 1 Approach(Selected) Y's-Joint KDE",
     "On-the-fly 1 Approach Y's-Joint KDE",
     "On-the-fly 2 Approach Y's-Joint KDE",
     "Control Approach Y's-Joint KDE",
     "Entropy 2 Approach(All) Y's-Joint KDE"])

```



8.5 Estimator of log-likelihood for the Parameters

The log likelihoods estimator will be computed with two fitting: The log likelihoods are computed for the ground truth values.

We compute the log likelihoods in two ways:

- David's estimator for pdf: by default this estimator calculates the pdf for each parameter separately. It is equivalent to the second fitting in Kde

- Kde estimator for pdf: For the kde we use two approaches. The first fitting will use all the samples for all the parameters at a given iteration

The second fitting will use only the data for one parameter to estimate a pdf of each parameter at a given iteration

David's Estimator for Parameters

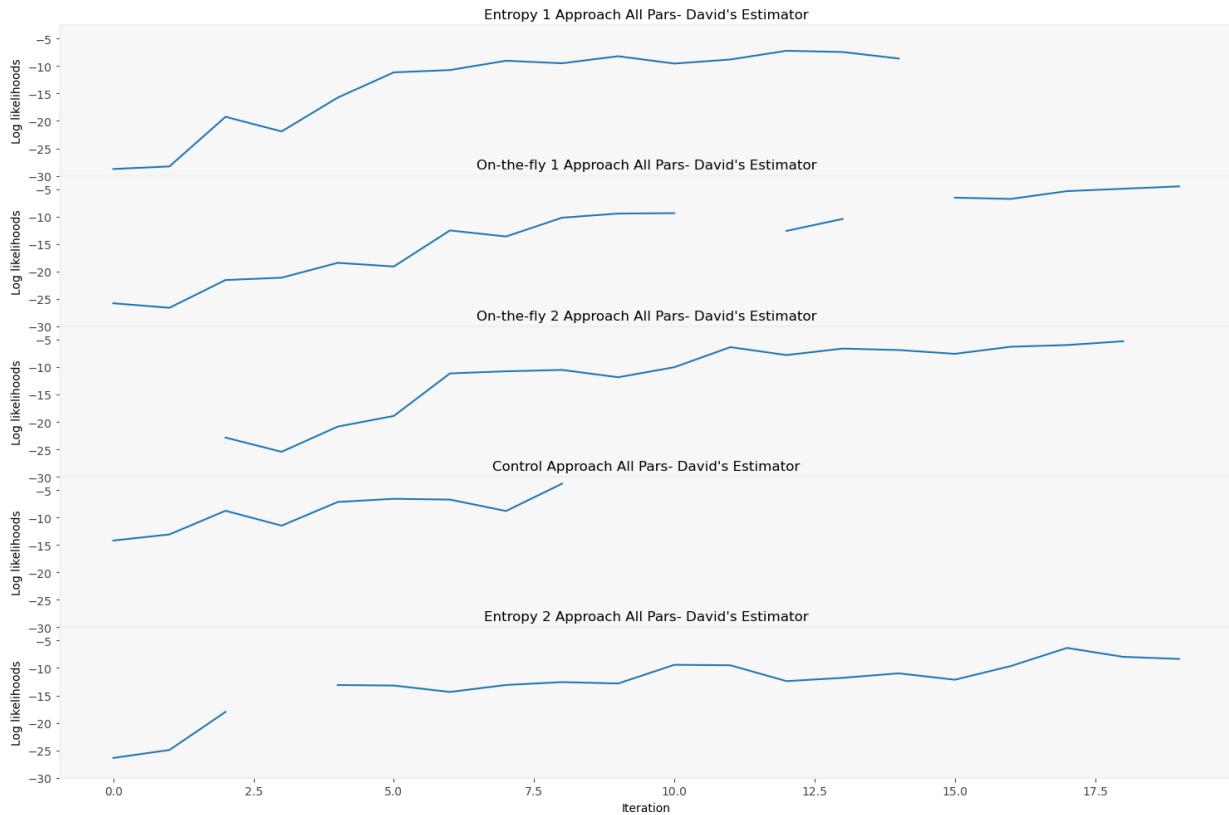
```
In [29]: #all_total_Likelihoods
Sum_likelihoods_at_each_iter_entropy1_for_pars = MyPlots.compute_likelihoods(exp_entropy1_for_pars, exp_entropy1_for_pars, 3)
Sum_likelihoods_at_each_iter_entropy2_for_pars = MyPlots.compute_likelihoods(exp_entropy2_for_pars, exp_entropy2_for_pars, 3)
Sum_likelihoods_at_each_iter_on_the_fly1_for_pars = MyPlots.compute_likelihoods(exp_on_the_fly1_for_pars, exp_on_the_fly1_for_pars, 3)
Sum_likelihoods_at_each_iter_on_the_fly2_for_pars = MyPlots.compute_likelihoods(exp_on_the_fly2_for_pars, exp_on_the_fly2_for_pars, 3)
Sum_likelihoods_at_each_iter_control_for_pars = MyPlots.compute_likelihoods(exp_control_for_pars, exp_control_for_pars, 3)
#Sum_likelihoods_at_each_iter_control_for_pars = compute_likelihoods(total_pts_control_for_pars, total_pts_control_for_pars, 3)

ERROR: the ground truth to estimate is too close to one extreme. So, we do not have enough samples either on the left or right of the ground truth
This is the left index
-3
this is right index
2
ERROR: the ground truth to estimate is too close to one extreme. So, we do not have enough samples either on the left or right of the ground truth
This is the left index
-2
this is right index
3
ERROR: the ground truth to estimate is too close to one extreme. So, we do not have enough samples either on the left or right of the ground truth
This is the left index
-2
this is right index
3
ERROR: the ground truth to estimate is too close to one extreme. So, we do not have enough samples either on the left or right of the ground truth
This is the left index
-3
this is right index
2
```

```
C:\Users\rober\FINAL NIST PROJECT\datastruct.py:353: VisibleDeprecationWarning: Creating an ndarray from ragged nested sequences (which is a list-or-tuple of lists-or-tuples-or ndarrays with different lengths or shapes) is deprecated. If you meant to do this, you must specify 'dtype=object' when creating the ndarray.
    return np.array(all_pts)
C:\Users\rober\anaconda3\lib\site-packages\numpy\core\fromnumeric.py:3432: RuntimeWarning: Mean of empty slice.
    return _methods._mean(a, axis=axis, dtype=dtype,
C:\Users\rober\anaconda3\lib\site-packages\numpy\core\_methods.py:190: RuntimeWarning: invalid value encountered in double_scalars
    ret = ret.dtype.type(ret / rcount)
```

In [30]:

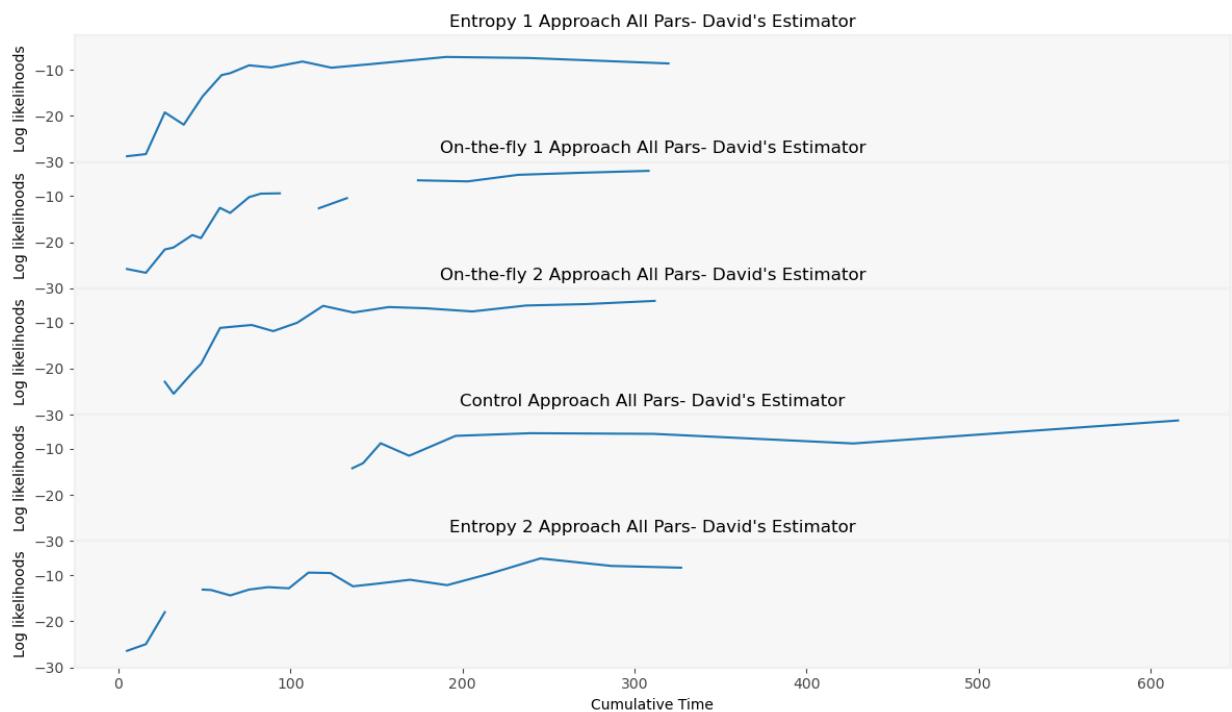
```
MyPlots.plot_likelihood_iterations(5,1,
    [Sum_likelihoods_at_each_iter_entropy1_for_pars,
     Sum_likelihoods_at_each_iter_on_the_fly1_for_pars,
     Sum_likelihoods_at_each_iter_on_the_fly2_for_pars,
     Sum_likelihoods_at_each_iter_control_for_pars,
     Sum_likelihoods_at_each_iter_entropy2_for_pars,
    ],
    ["Entropy 1 Approach All Pars- David's Estimator",
     "On-the-fly 1 Approach All Pars- David's Estimator",
     "On-the-fly 2 Approach All Pars- David's Estimator",
     "Control Approach All Pars- David's Estimator",
     "Entropy 2 Approach All Pars- David's Estimator"
    ])
#help(plot_Likelihood_iterations)
```



In [31]:

```
MyPlots.plot_likelihood_time(5,1,
    [Sum_likelihoods_at_each_iter_entropy1_for_pars,
     Sum_likelihoods_at_each_iter_on_the_fly1_for_pars,
     Sum_likelihoods_at_each_iter_on_the_fly2_for_pars,
     Sum_likelihoods_at_each_iter_control_for_pars,
     Sum_likelihoods_at_each_iter_entropy2_for_pars]
```

```
[],
[ exp_entropy1.totaltimes(),
exp_on_the_fly1.totaltimes(),
exp_on_the_fly2.totaltimes(),
exp_control.totaltimes(),
exp_entropy2.totaltimes()
],
["Entropy 1 Approach All Pars- David's Estimator",
"On-the-fly 1 Approach All Pars- David's Estimator",
"On-the-fly 2 Approach All Pars- David's Estimator",
"Control Approach All Pars- David's Estimator",
"Entropy 2 Approach All Pars- David's Estimator"
])
```



Multidimensional KDE Estimator for Parameters

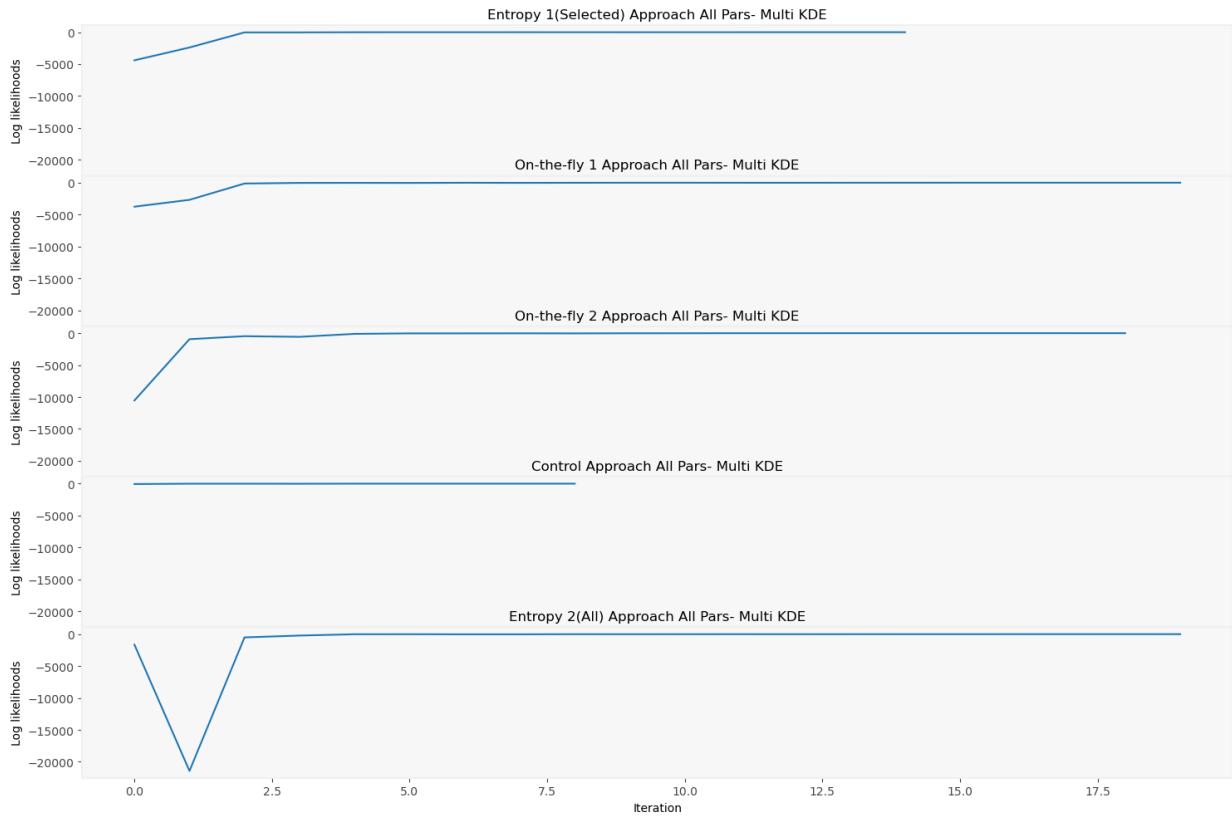
Combined Parameters

```
In [32]: #all_total_Likelihoods
Sum_likelihoods_at_each_iter_entropy1_for_pars_kde = MyPlots.likelihood_helper_kde(exp_
exp_entropy1.get_t
Sum_likelihoods_at_each_iter_entropy2_for_pars_kde = MyPlots.likelihood_helper_kde(exp_
exp_
Sum_likelihoods_at_each_iter_on_the_fly1_for_pars_kde = MyPlots.likelihood_helper_kde(exp_
exp_on_the_
Sum_likelihoods_at_each_iter_on_the_fly2_for_pars_kde = MyPlots.likelihood_helper_kde(exp_
exp_on_the_
Sum_likelihoods_at_each_iter_control_for_pars_kde = MyPlots.likelihood_helper_kde(exp_
exp_on_the_
```

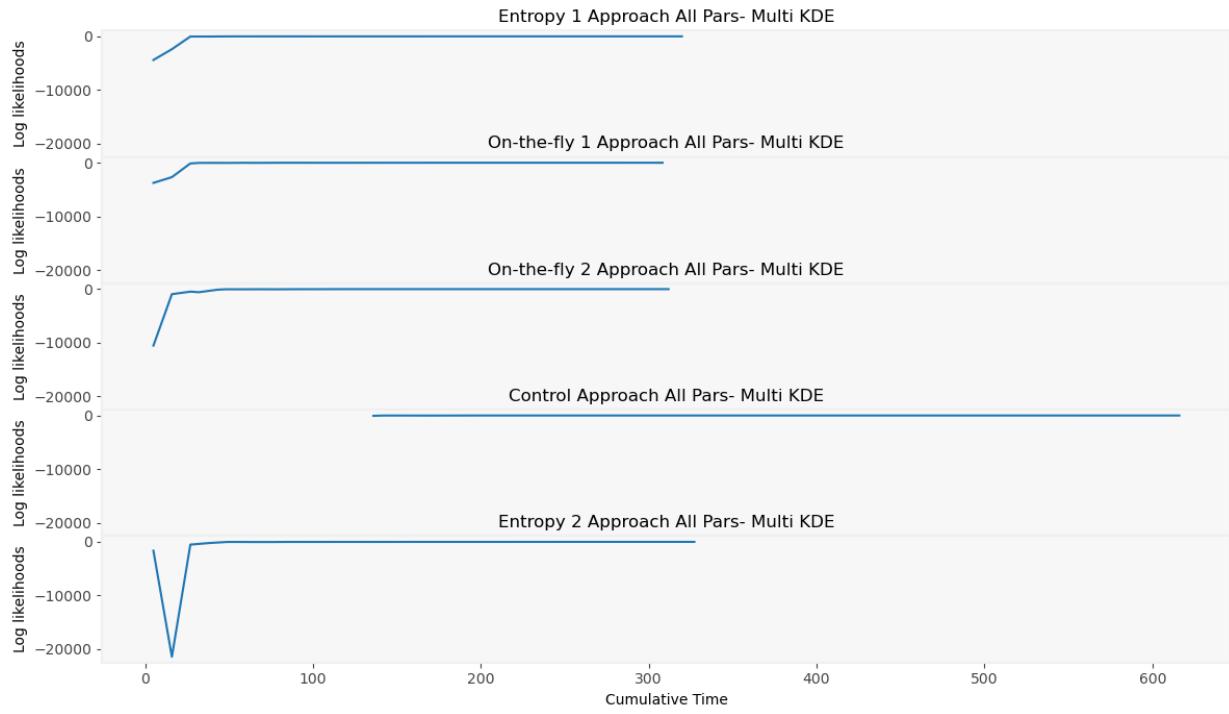
```
C:\Users\rober\FINAL NIST PROJECT\datastruct.py:353: VisibleDeprecationWarning: Creating an ndarray from ragged nested sequences (which is a list-or-tuple of lists-or-tuples-or ndarrays with different lengths or shapes) is deprecated. If you meant to do this, you must specify 'dtype=object' when creating the ndarray.  
    return np.array(all_pts)  
C:\Users\rober\FINAL NIST PROJECT\datastruct.py:353: VisibleDeprecationWarning: Creating an ndarray from ragged nested sequences (which is a list-or-tuple of lists-or-tuples-or ndarrays with different lengths or shapes) is deprecated. If you meant to do this, you must specify 'dtype=object' when creating the ndarray.  
    return np.array(all_pts)  
C:\Users\rober\FINAL NIST PROJECT\datastruct.py:353: VisibleDeprecationWarning: Creating an ndarray from ragged nested sequences (which is a list-or-tuple of lists-or-tuples-or ndarrays with different lengths or shapes) is deprecated. If you meant to do this, you must specify 'dtype=object' when creating the ndarray.  
    return np.array(all_pts)  
C:\Users\rober\FINAL NIST PROJECT\datastruct.py:353: VisibleDeprecationWarning: Creating an ndarray from ragged nested sequences (which is a list-or-tuple of lists-or-tuples-or ndarrays with different lengths or shapes) is deprecated. If you meant to do this, you must specify 'dtype=object' when creating the ndarray.  
    return np.array(all_pts)
```

```
In [33]: MyPlots.plot_likelihood_iterations(5,1,  
                                         [Sum_likelihoods_at_each_iter_entropy1_for_pars_kde,  
                                          Sum_likelihoods_at_each_iter_on_the_fly1_for_pars_kde,  
                                          Sum_likelihoods_at_each_iter_on_the_fly2_for_pars_kde,  
                                          Sum_likelihoods_at_each_iter_control_for_pars_kde,  
                                          Sum_likelihoods_at_each_iter_entropy2_for_pars_kde  
                                         ]  
,["Entropy 1(Selected) Approach All Pars- Multi KDE",  
 "On-the-fly 1 Approach All Pars- Multi KDE",  
 "On-the-fly 2 Approach All Pars- Multi KDE",  
 "Control Approach All Pars- Multi KDE",  
 "Entropy 2(All) Approach All Pars- Multi KDE"])
```

Run Plots



```
In [34]: MyPlots.plot_likelihood_time(5,1,
    [Sum_likelihoods_at_each_iter_entropy1_for_pars_kde,
     Sum_likelihoods_at_each_iter_on_the_fly1_for_pars_kde,
     Sum_likelihoods_at_each_iter_on_the_fly2_for_pars_kde,
     Sum_likelihoods_at_each_iter_control_for_pars_kde,
     Sum_likelihoods_at_each_iter_entropy2_for_pars_kde
    ],
    [exp_entropy1.totaltimes(),
     exp_on_the_fly1.totaltimes(),
     exp_on_the_fly2.totaltimes(),
     exp_control.totaltimes(),
     exp_entropy2.totaltimes()
    ],
    ["Entropy 1 Approach All Pars- Multi KDE",
     "On-the-fly 1 Approach All Pars- Multi KDE",
     "On-the-fly 2 Approach All Pars- Multi KDE",
     "Control Approach All Pars- Multi KDE",
     "Entropy 2 Approach All Pars- Multi KDE"
    ])
)
```



Separated for each parameter. We will use the kde in the data for each parameter to build independent pdf's for each parameter

```
In [35]: #Example with entropy approach
kdes_entropy1, log_pars_entropy1 = MyPlots.kdes_and_loglikelihoods_for_pars(exp_entropy1.get_true())
exp_entropy1.get_true()

kdes_entropy2, log_pars_entropy2 = MyPlots.kdes_and_loglikelihoods_for_pars(exp_entropy2.get_true())
exp_entropy2.get_true()

#Example with on-the-fly approach
kdes_on_the_fly1, log_pars_on_the_fly1 = MyPlots.kdes_and_loglikelihoods_for_pars(exp_on_the_fly1.get_true())
exp_on_the_fly1.get_true()

kdes_on_the_fly2, log_pars_on_the_fly2 = MyPlots.kdes_and_loglikelihoods_for_pars(exp_on_the_fly2.get_true())
exp_on_the_fly2.get_true()

kdes_control, log_pars_control = MyPlots.kdes_and_loglikelihoods_for_pars(exp_control.get_true())
exp_control.get_true()

#####
#Comment out the section below. It makes sense only if you have a GMM version
#####

## #Example with entropy approach
# kdes_entropy1_gmm, log_pars_entropy1_gmm = MyPlots.kdes_and_LogLikelihoods_for_pars(exp_entropy1_gmm.get_true())
# exp_entropy1_gmm.get_true()

# kdes_entropy2_gmm, log_pars_entropy2_gmm = MyPlots.kdes_and_LogLikelihoods_for_pars(exp_entropy2_gmm.get_true())
# exp_entropy2_gmm.get_true()

## #Example with on-the-fly approach
# kdes_on_the_fly1_gmm, log_pars_on_the_fly1_gmm = MyPlots.kdes_and_LogLikelihoods_for_pars(exp_on_the_fly1_gmm.get_true())
# exp_on_the_fly1_gmm.get_true()

# kdes_on_the_fly2_gmm, log_pars_on_the_fly2_gmm = MyPlots.kdes_and_LogLikelihoods_for_pars(exp_on_the_fly2_gmm.get_true())
# exp_on_the_fly2_gmm.get_true()

## kdes_control_gmm, log_pars_control_gmm = MyPlots.kdes_and_LogLikelihoods_for_pars(exp_control_gmm.get_true())
# exp_control_gmm.get_true()
```

```
C:\Users\rober\FINAL NIST PROJECT\datastruct.py:353: VisibleDeprecationWarning: Creating an ndarray from ragged nested sequences (which is a list-or-tuple of lists-or-tuples-or ndarrays with different lengths or shapes) is deprecated. If you meant to do this, you must specify 'dtype=object' when creating the ndarray.
    return np.array(all_pts)
C:\Users\rober\FINAL NIST PROJECT\datastruct.py:353: VisibleDeprecationWarning: Creating an ndarray from ragged nested sequences (which is a list-or-tuple of lists-or-tuples-or ndarrays with different lengths or shapes) is deprecated. If you meant to do this, you must specify 'dtype=object' when creating the ndarray.
    return np.array(all_pts)
C:\Users\rober\FINAL NIST PROJECT\datastruct.py:353: VisibleDeprecationWarning: Creating an ndarray from ragged nested sequences (which is a list-or-tuple of lists-or-tuples-or ndarrays with different lengths or shapes) is deprecated. If you meant to do this, you must specify 'dtype=object' when creating the ndarray.
    return np.array(all_pts)
C:\Users\rober\FINAL NIST PROJECT\datastruct.py:353: VisibleDeprecationWarning: Creating an ndarray from ragged nested sequences (which is a list-or-tuple of lists-or-tuples-or ndarrays with different lengths or shapes) is deprecated. If you meant to do this, you must specify 'dtype=object' when creating the ndarray.
    return np.array(all_pts)
C:\Users\rober\FINAL NIST PROJECT\datastruct.py:353: VisibleDeprecationWarning: Creating an ndarray from ragged nested sequences (which is a list-or-tuple of lists-or-tuples-or ndarrays with different lengths or shapes) is deprecated. If you meant to do this, you must specify 'dtype=object' when creating the ndarray.
    return np.array(all_pts)
```

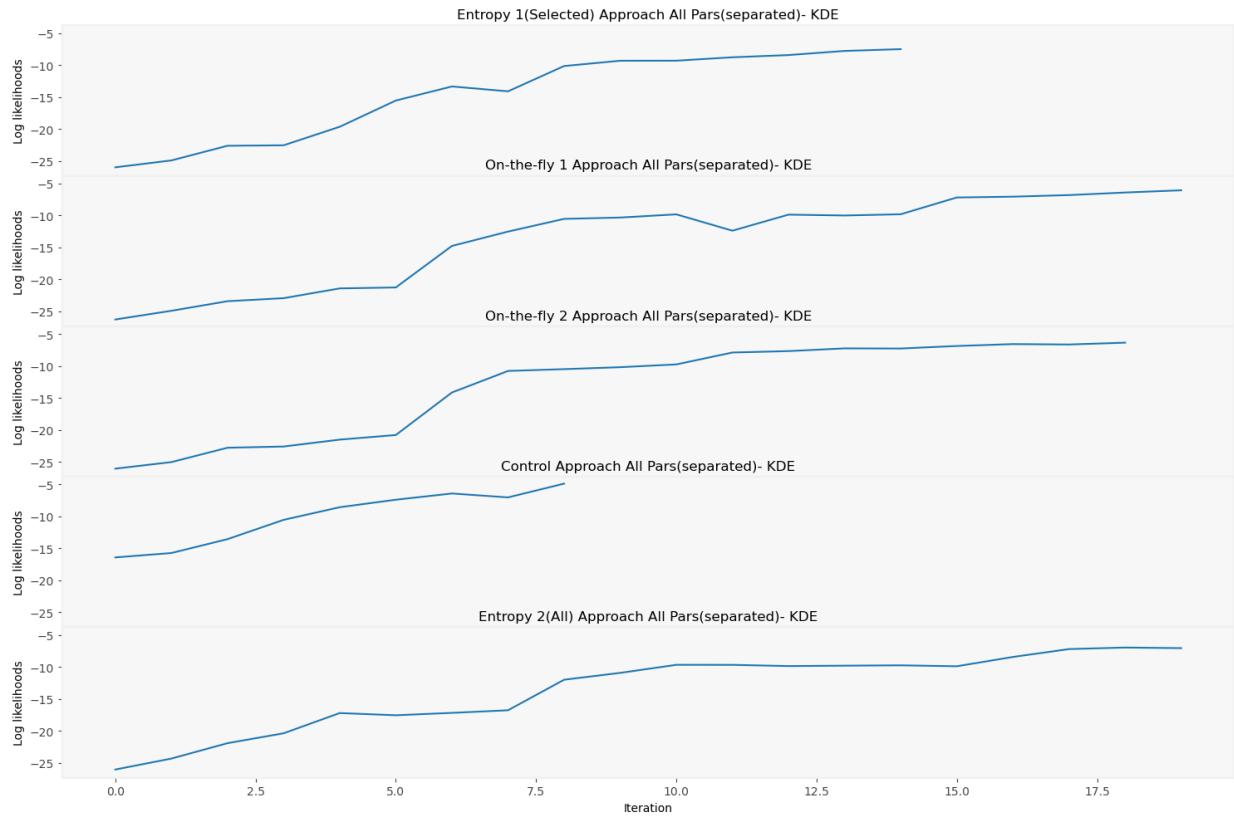
In [36]:

```
#Adding the loglikelihhods of the parameters at a given iteration
log_sums_kde_separated_iters_entropy1 = np.sum(log_pars_entropy1, axis= 1)
log_sums_kde_separated_iters_entropy2 = np.sum(log_pars_entropy2, axis= 1)
log_sums_kde_sepataed_iters_on_the_fly1 = np.sum(log_pars_on_the_fly1, axis= 1)
log_sums_kde_sepataed_iters_on_the_fly2 = np.sum(log_pars_on_the_fly2, axis= 1)
log_sums_kde_separated_iters_control = np.sum(log_pars_control, axis= 1)
```

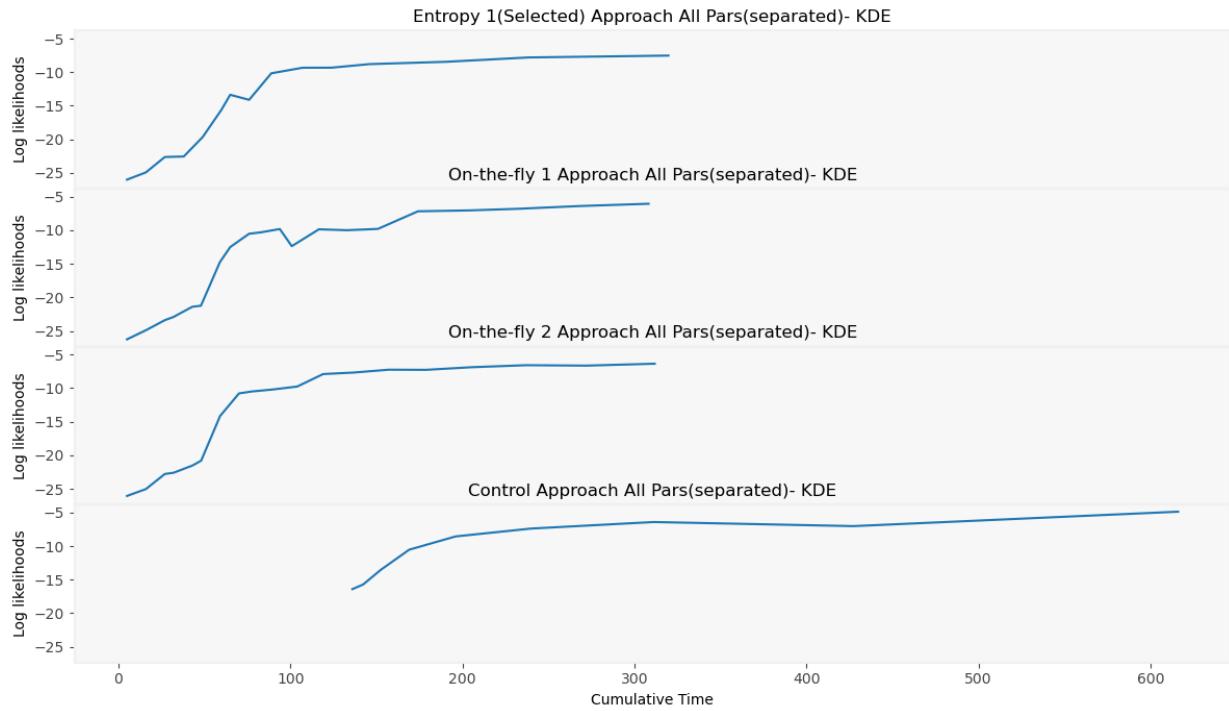
In [37]:

```
MyPlots.plot_likelihood_iterations(5,1,
                                    [log_sums_kde_separated_iters_entropy1,
                                     log_sums_kde_sepataed_iters_on_the_fly1,
                                     log_sums_kde_sepataed_iters_on_the_fly2 ,
                                     log_sums_kde_separated_iters_control,
                                     log_sums_kde_separated_iters_entropy2],
                                    ["Entropy 1(Selected) Approach All Pars(separated)- KDE",
                                     "On-the-fly 1 Approach All Pars(separated)- KDE",
                                     "On-the-fly 2 Approach All Pars(separated)- KDE",
                                     "Control Approach All Pars(separated)- KDE",
                                     "Entropy 2(All) Approach All Pars(separated)- KDE"])
```

Run Plots



```
In [38]: MyPlots.plot_likelihood_time(4,1,
    [log_sums_kde_separated_iters_entropy1,
     log_sums_kde_separaated_iters_on_the_fly1,
     log_sums_kde_sepataed_iters_on_the_fly2,
     log_sums_kde_separated_iters_control,
     log_sums_kde_separated_iters_entropy2],
    [exp_entropy1.totaltimes(),
     exp_on_the_fly1.totaltimes(),
     exp_on_the_fly2.totaltimes(),
     exp_control.totaltimes(),
     exp_entropy2.totaltimes()],
    ["Entropy 1(Selected) Approach All Pars(separated)- KDE",
     "On-the-fly 1 Approach All Pars(separated)- KDE",
     "On-the-fly 2 Approach All Pars(separated)- KDE",
     "Control Approach All Pars(separated)- KDE",
     "Entropy 2(All) Approach All Pars(separated)- KDE"]
    ])
```



Histogram for Parameter Samples

A

1. First we look at the last sample for all the autonomous approaches.
2. Second, we look at the result using GMM

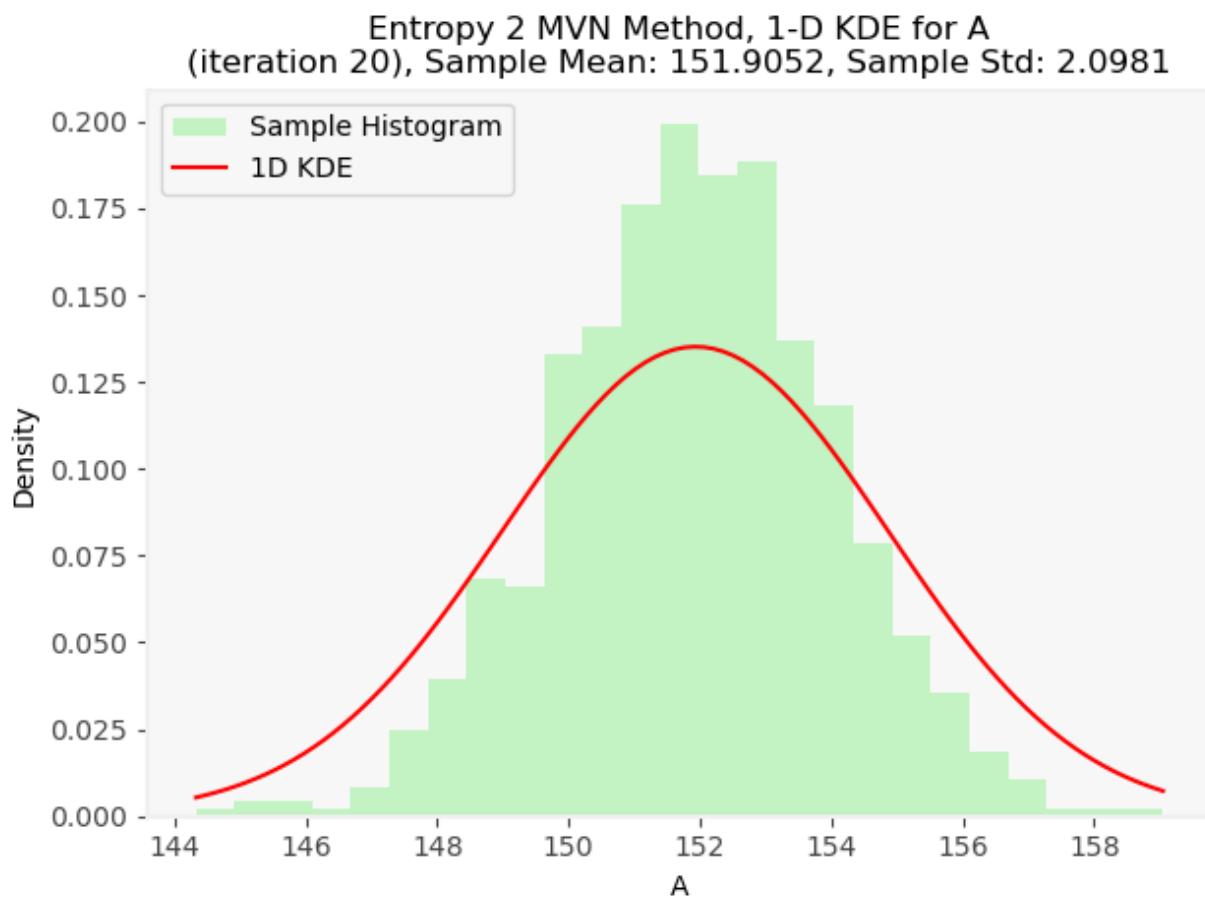
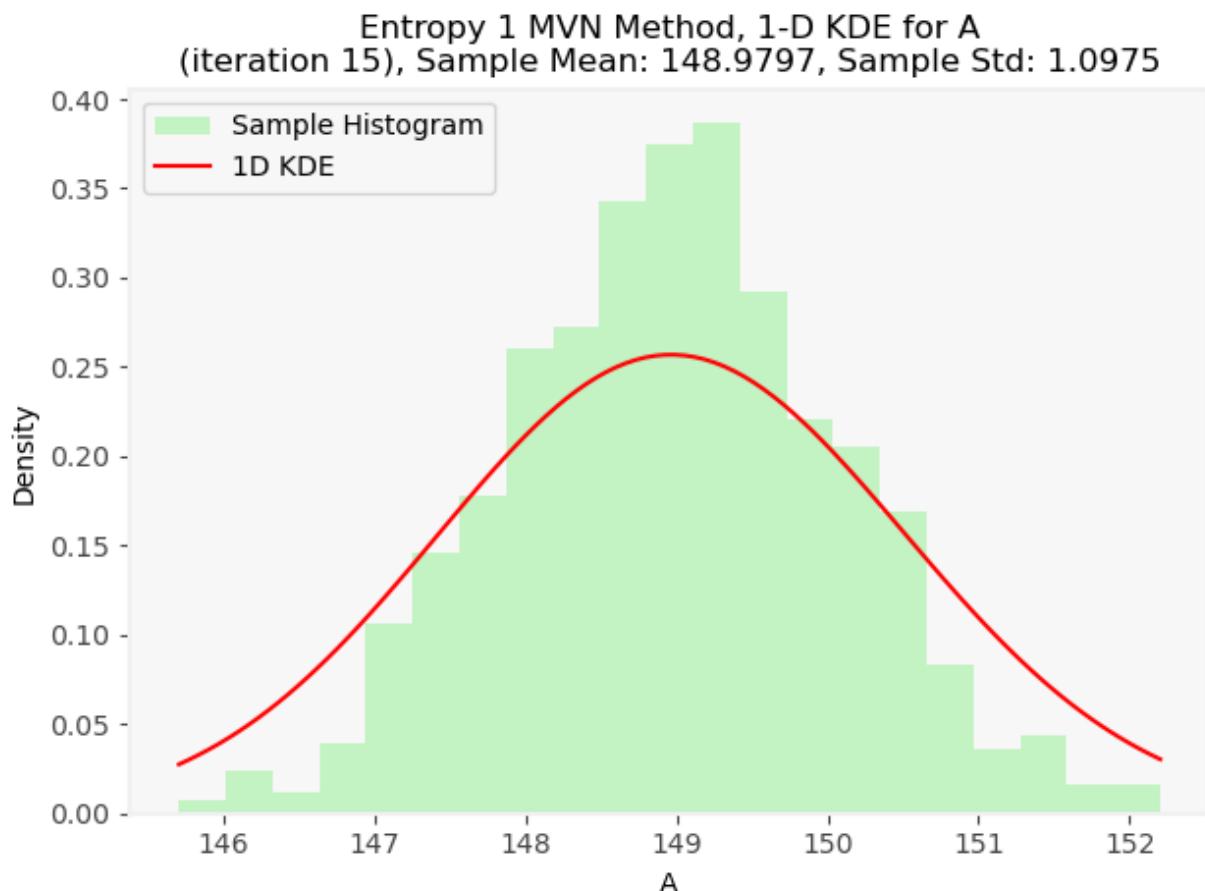
```
In [39]: MyPlots.plot_hist_1d_kde(list_par_separated_e1[0][-1], kdes_entropy1[-1,0],"Entropy 1"
plt.show()

MyPlots.plot_hist_1d_kde(list_par_separated_e2[0][-1], kdes_entropy2[-1,0],"Entropy 2"
len(exp_entropy2.totaltimes()))
plt.show()

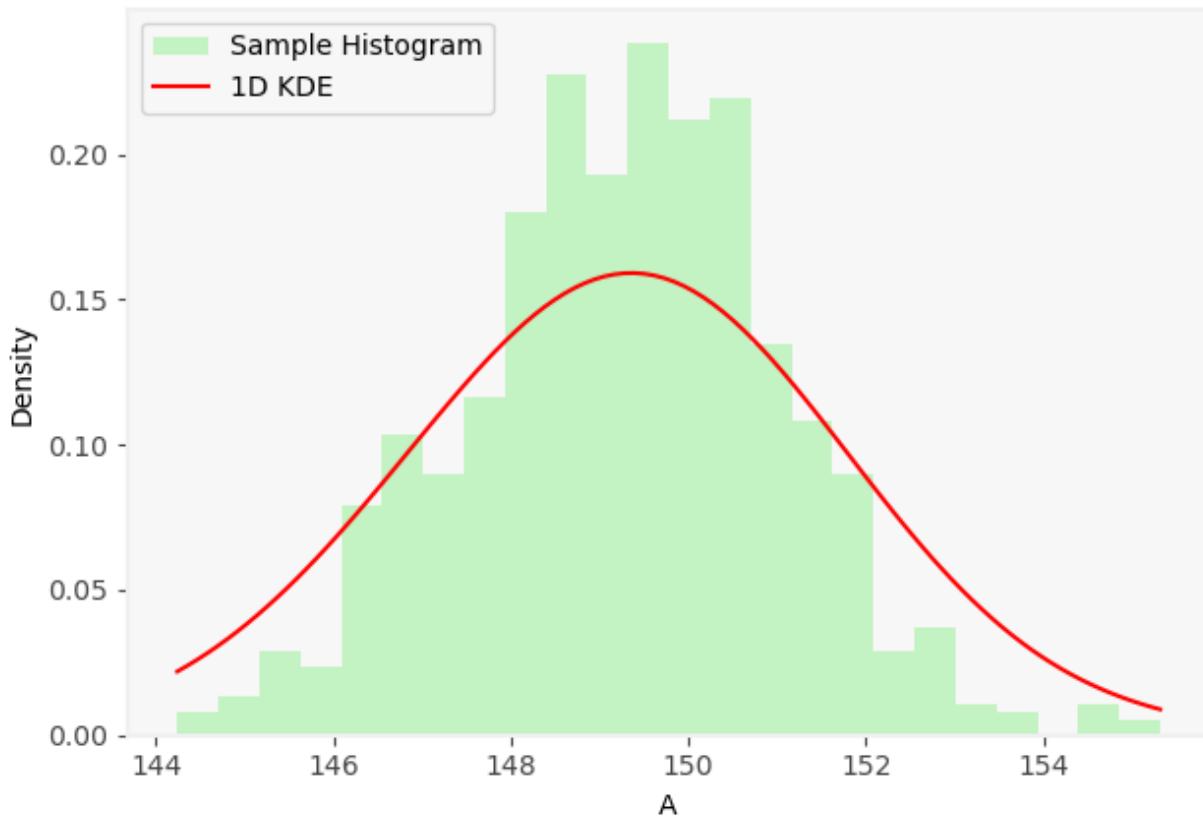
MyPlots.plot_hist_1d_kde(list_par_separated_o1[0][-1], kdes_on_the_fly1[-1,0],"On-the-
len(exp_on_the_fly1.totaltimes())
plt.show()

MyPlots.plot_hist_1d_kde(list_par_separated_o2[0][-1], kdes_on_the_fly2[-1,0],"On-the-
len(exp_on_the_fly2.totaltimes())
plt.show()

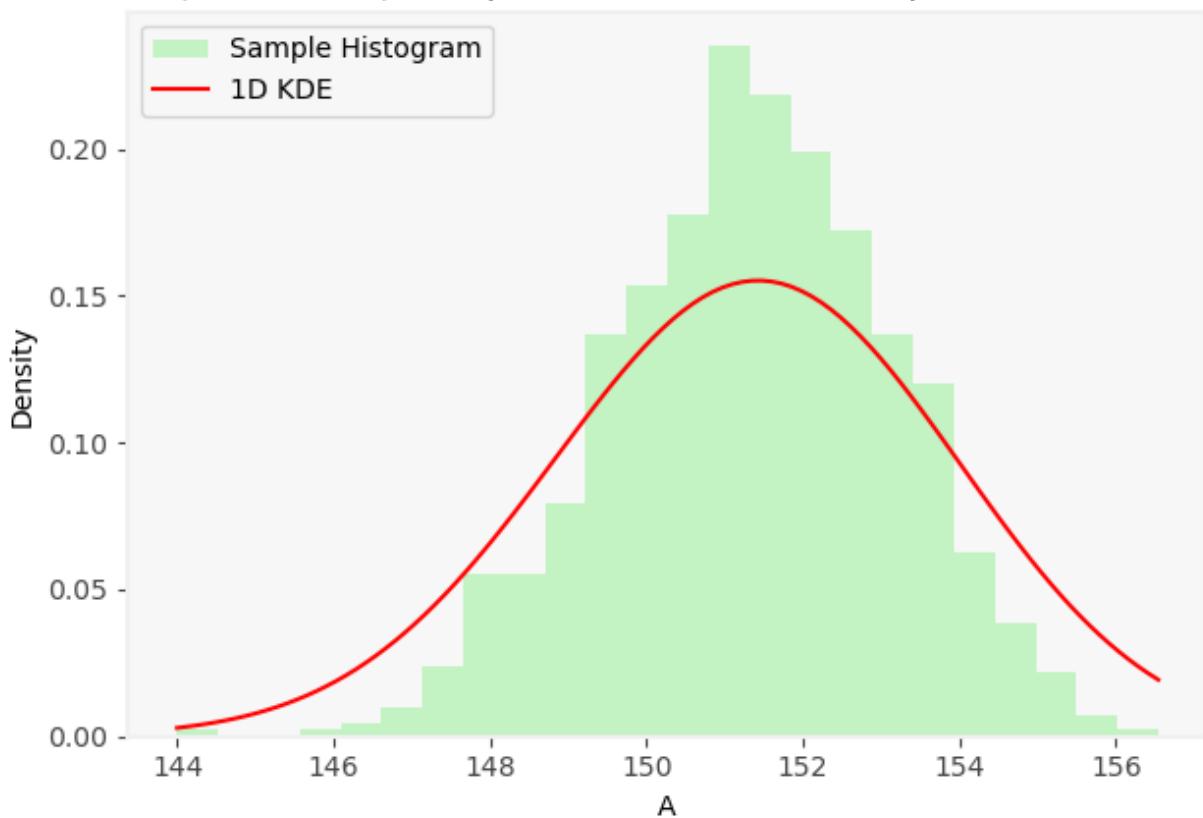
MyPlots.plot_hist_1d_kde(list_par_separated_c[0][-1], kdes_control[-1,0],"Control MVN"
len(exp_control.totaltimes()))
plt.show()
```

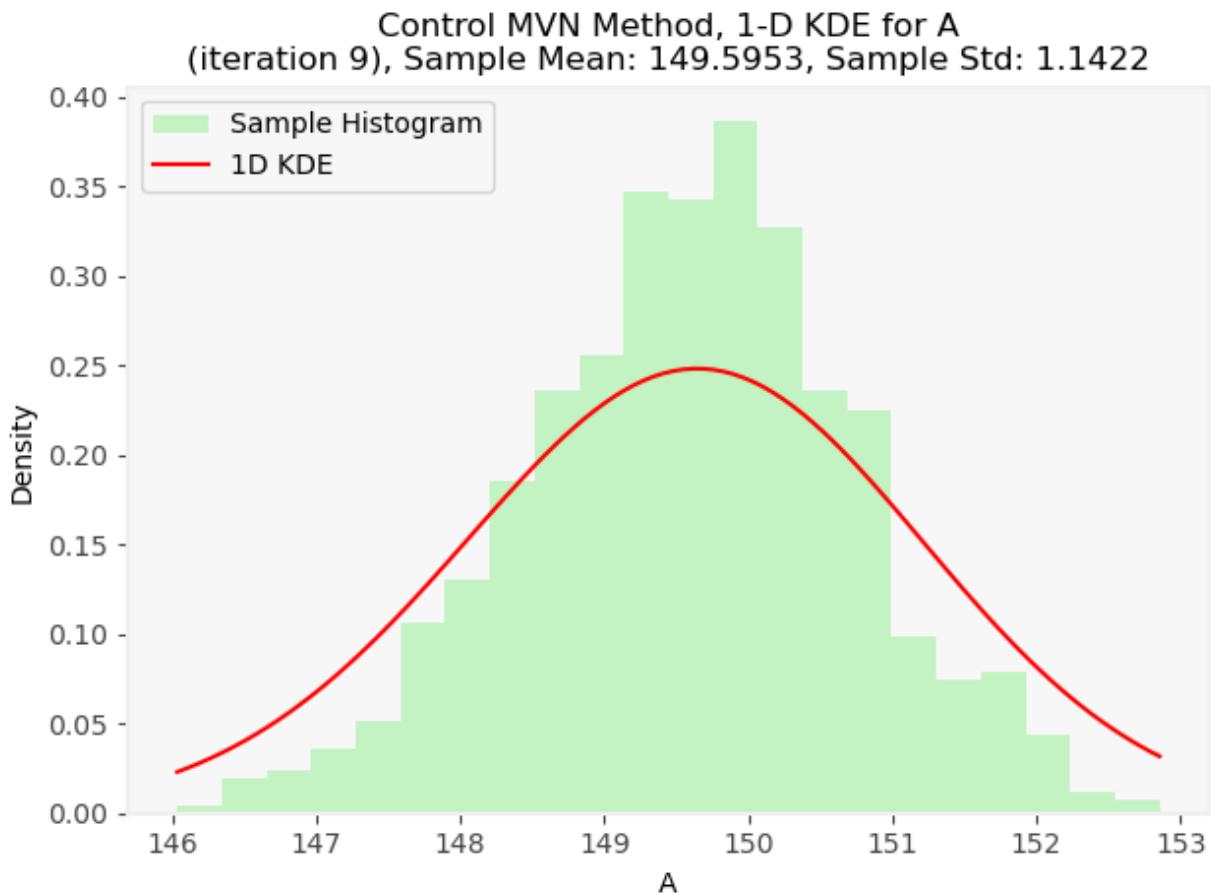


On-the-fly 1 MVN Method, 1-D KDE for A
(iteration 20), Sample Mean: 149.2862, Sample Std: 1.7829



On-the-fly 2 MVN Method, 1-D KDE for A
(iteration 19), Sample Mean: 151.3382, Sample Std: 1.8125





```
In [40]: #DELETE THIS LATER THIS MAKES SENSE ONLY IF YOU HAVE ALSO A GMM VERSION #####
```

```
# MyPlots.plot_hist_1d_kde(list_par_separated_e1_gmm[0][-1],
#                           kdes_entropy1_gmm[-1,0], "Entropy 1 GMM", "A",
#                           len(exp_entropy1_gmm.totaltimes()))
# plt.show()

# MyPlots.plot_hist_1d_kde(list_par_separated_e2_gmm[0][-1], kdes_entropy2_gmm[-1,0],
#                           len(exp_entropy2_gmm.totaltimes()))
# plt.show()

# MyPlots.plot_hist_1d_kde(list_par_separated_o1_gmm[0][-1], kdes_on_the_fly1_gmm[-1,0],
#                           len(exp_on_the_fly1_gmm.totaltimes()))
# plt.show()

# MyPlots.plot_hist_1d_kde(list_par_separated_o2_gmm[0][-1], kdes_on_the_fly2_gmm[-1,0],
#                           len(exp_on_the_fly2_gmm.totaltimes()))
# plt.show()

# MyPlots.plot_hist_1d_kde(list_par_separated_c_gmm[0][-1], kdes_control_gmm[-1,0], "Control GMM", "A",
#                           len(exp_control_gmm.totaltimes()))
```

```
#####
#####
```

Recalculating Entropies for the cases using the GMM

```
In [41]: # entropy1_H_total_gmm = recalculating_entropy(exp_entropy1, None, gmm_setting)
# entropy1_H_marg_gmm = recalculating_entropy(exp_entropy1, exp_entropy1.settings.sel,
#                                              gmm_setting)

# entropy2_H_total_gmm = recalculating_entropy(exp_entropy2, None, gmm_setting)
# entropy2_H_marg_gmm = recalculating_entropy(exp_entropy2, exp_on_the_fly1.settings.sel,
#                                              gmm_setting)

# on_the_fly1_H_total_gmm = recalculating_entropy(exp_on_the_fly1, None, gmm_setting)
# on_the_fly1_H_marg_gmm = recalculating_entropy(exp_on_the_fly1, exp_on_the_fly1.settings.sel,
#                                              gmm_setting)

# on_the_fly2_H_total_gmm = recalculating_entropy(exp_on_the_fly2, None, gmm_setting)
# on_the_fly2_H_marg_gmm = recalculating_entropy(exp_on_the_fly2, exp_on_the_fly2.settings.sel,
#                                              gmm_setting)

# control_H_total_gmm = recalculating_entropy(exp_control, None, gmm_setting)
# control_H_marg_gmm = recalculating_entropy(exp_control, exp_control.settings.sel, gmm_setting)
```

```
In [42]: #Notice we could also print the total entropy, instead, of the marginalized entropy.
#On-thefly always deals with the total entropy. You cannot choose a parameter of interest.

#Using the recalculated entropies using GMM

# times = [exp_entropy1.totaltimes(), exp_on_the_fly1.totaltimes(),
#           exp_on_the_fly2.totaltimes(), exp_control.totaltimes(), exp_entropy2.totaltimes()]
# entropies = [entropy1_H_total_gmm, on_the_fly1_H_total_gmm,
#              on_the_fly2_H_total_gmm, control_H_total_gmm, entropy2_H_total_gmm]
# MyPlots.plot_entropy_times(times, entropies)

#C1 blue entropy
#C2 yellow on the fly
#C3 green on the fly
#C4 red control
#C5 Purple entropy2

#Total Entropy
```

```
In [43]: #####This works only if you have a GMM version

# MyPlots.plot_entropy_times([exp_entropy1.totaltimes(),exp_entropy1.totaltimes()],
#                            [exp_entropy1.entropy(), entropy1_H_total_gmm])

#Blue MVN
#Orange GMM
```

```
In [44]: # MyPlots.plot_entropy_times([exp_entropy2.totaltimes(),exp_entropy2.totaltimes()],
#                            [exp_entropy2.entropy(), entropy2_H_total_gmm])
```

```
#Again undistinguishable
```

```
In [45]: # MyPlots.plot_entropy_times([exp_on_the_fly1.totaltimes(),exp_on_the_fly1.totaltimes()
#                                         [exp_on_the_fly1.entropy(), on_the_fly1_H_total_gmm])
```

```
#Again undistinguishable
```

```
In [46]: # MyPlots.plot_entropy_times([exp_on_the_fly2.totaltimes(),exp_on_the_fly2.totaltimes()
#                                         [exp_on_the_fly2.entropy(), on_the_fly2_H_total_gmm])
```

```
#Again undistinguishable
```

```
In [47]: #Notice we could also print the total entropy, instead, of the marginalized entropy.
#On-thefly always deals with the total entropy. You cannot choose a parameter of inter
```

```
# times = [exp_entropy1.totaltimes(), exp_on_the_fly1.totaltimes(),
#           exp_on_the_fly2.totaltimes(),exp_control.totaltimes(), exp_entropy2.totaltime
# entropies = [entropy1_H_marg_gmm, on_the_fly1_H_marg_gmm,
#              on_the_fly2_H_marg_gmm, control_H_marg_gmm, entropy2_H_marg_gmm]
# MyPlots.plot_entropy_times(times, entropies)
```

```
In [48]: # MyPlots.plot_entropy_times([exp_entropy1.totaltimes(),exp_entropy1.totaltimes()],
#                               [exp_entropy1.entropy_marg(),entropy1_H_marg_gmm ])
```

```
In [49]: # MyPlots.plot_entropy_times([exp_entropy2.totaltimes(),exp_entropy2.totaltimes()],
#                               [exp_entropy2.entropy_marg(),entropy2_H_marg_gmm ])
```

```
In [50]: # MyPlots.plot_entropy_times([exp_on_the_fly1.totaltimes(),exp_on_the_fly1.totaltimes(),
#                               [exp_on_the_fly1.entropy_marg(),on_the_fly1_H_marg_gmm])
```

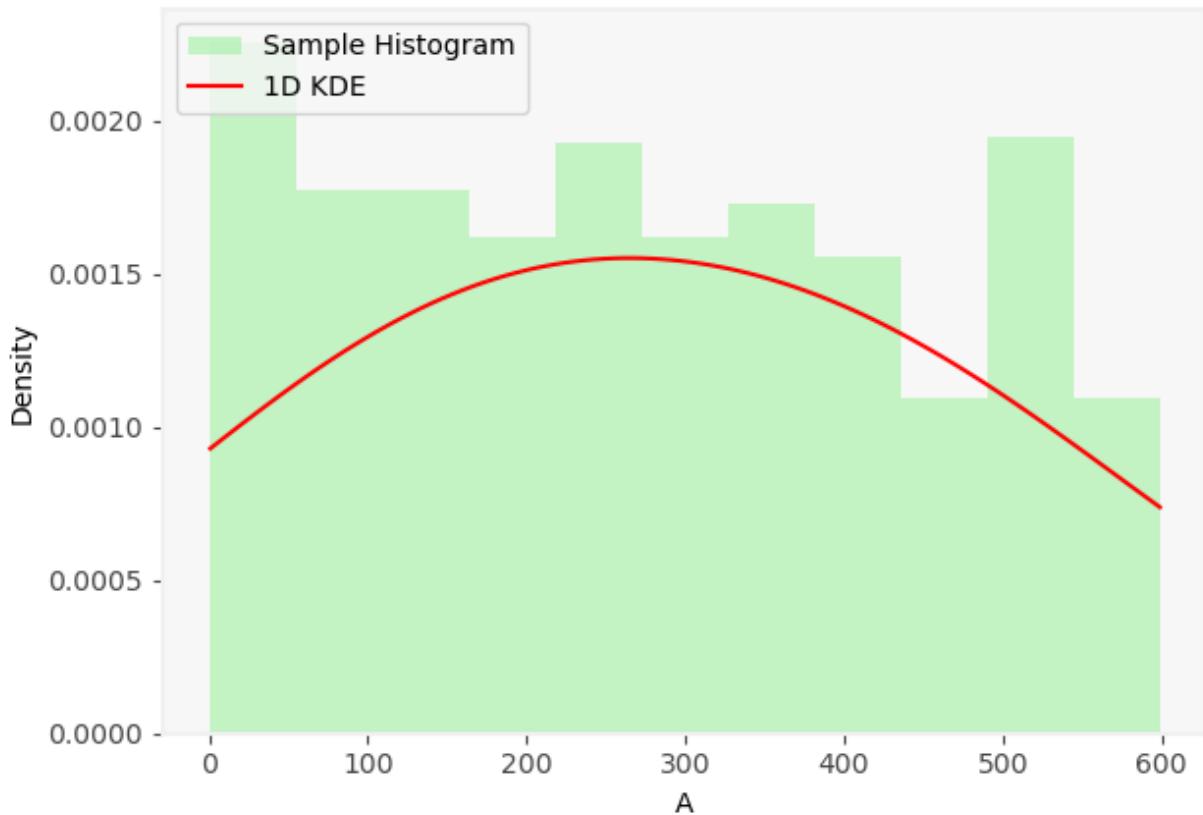
```
In [51]: # MyPlots.plot_entropy_times([exp_on_the_fly2.totaltimes(),exp_on_the_fly2.totaltimes(),
#                               [exp_on_the_fly2.entropy_marg(),on_the_fly2_H_marg_gmm])
```

End of the Recalculation section

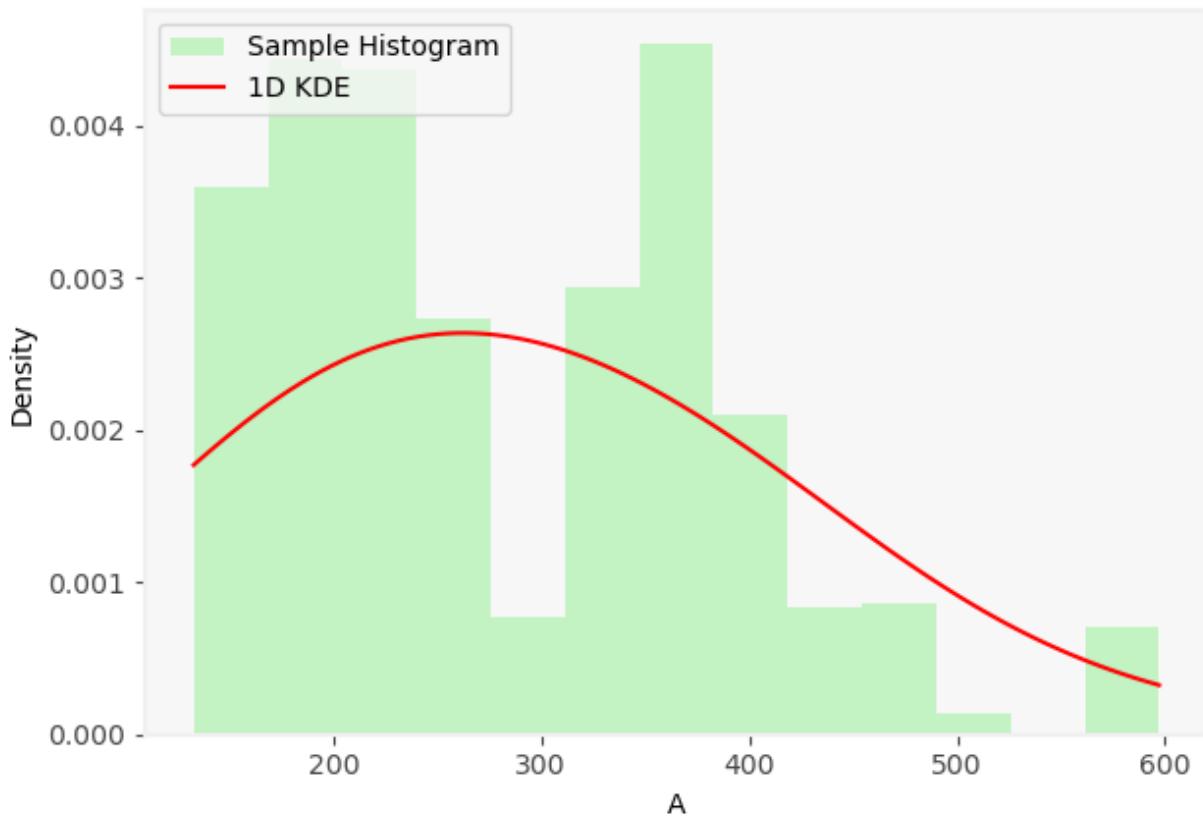
```
In [52]: #Printing the evolution of parameter I for Entropy
#Later do the same for parameter A since that was the parameter selected by entropy
```

```
for i in range(len(exp_entropy1.totaltimes())):
    MyPlots.plot_hist_1d_kde(list_par_separated_e1[0][i], kdes_entropy1[i,0],"Entropy
```

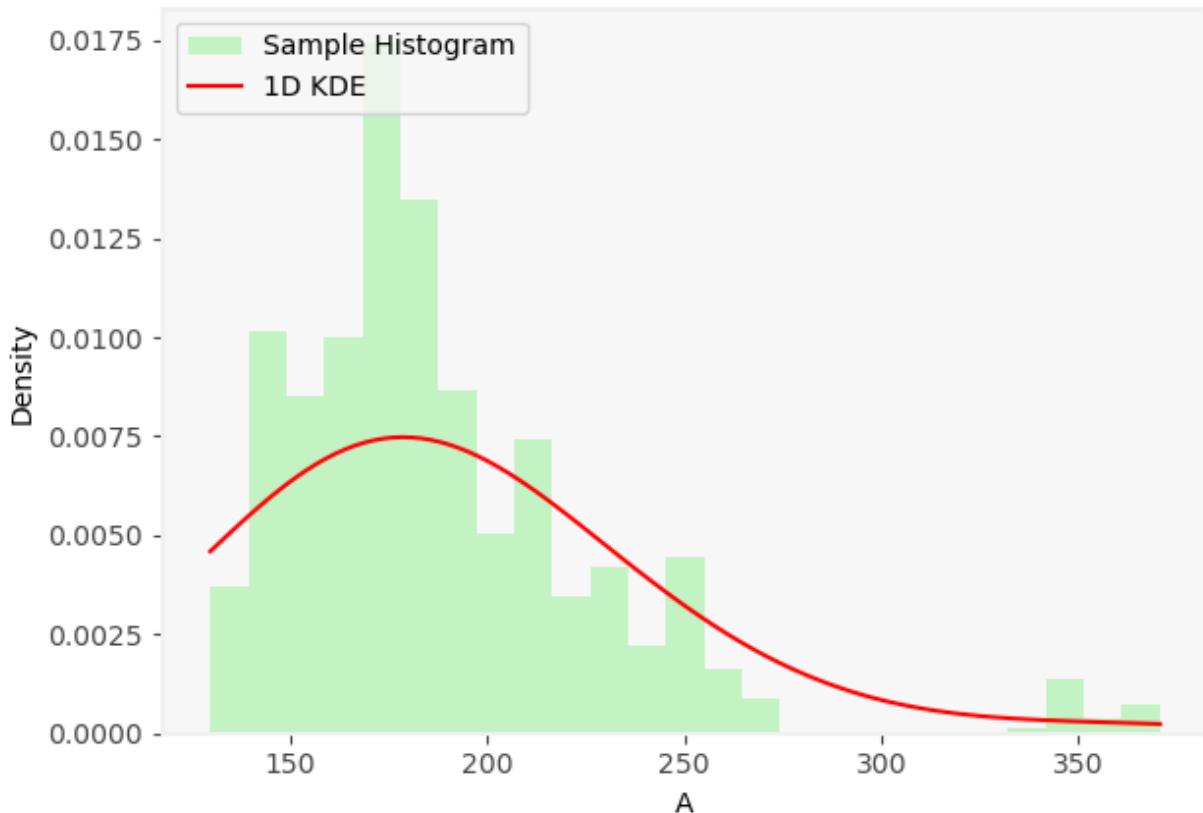
Entropy 1 Method, 1-D KDE for A
(iteration 0), Sample Mean: 277.8228, Sample Std: 170.6427



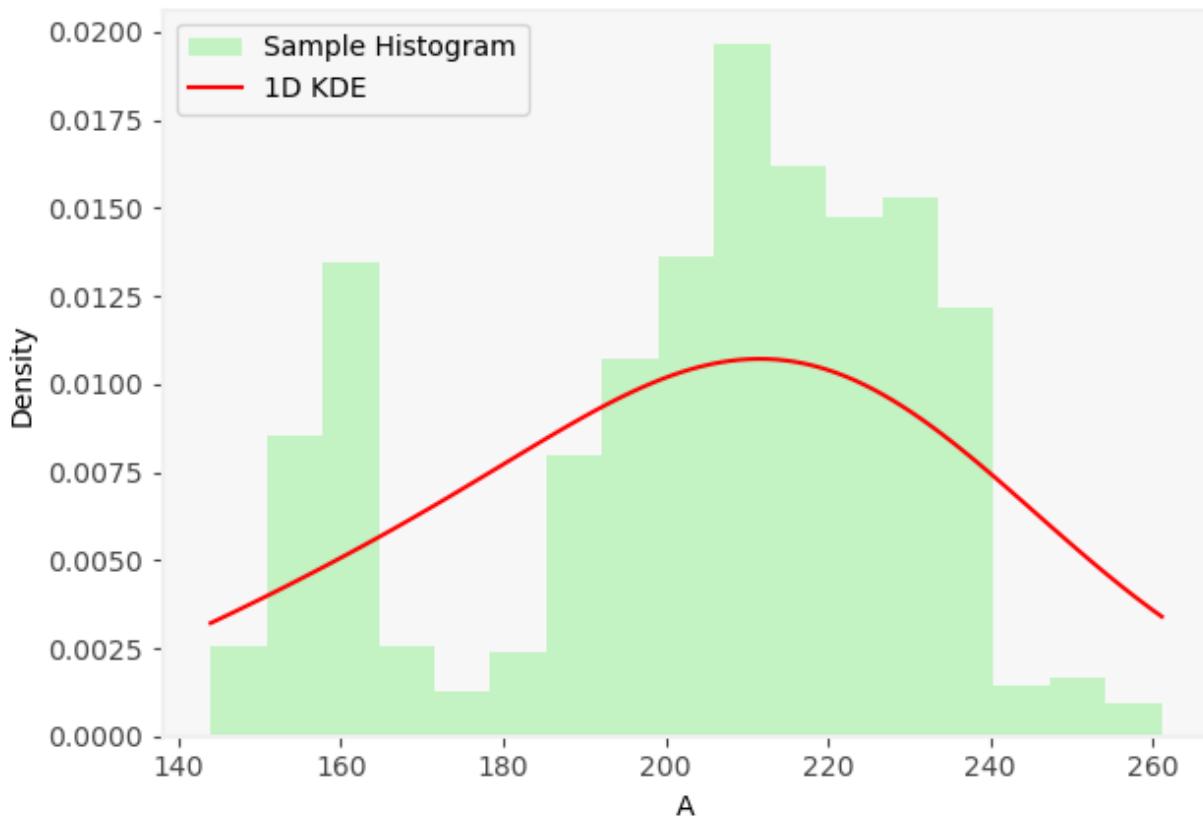
Entropy 1 Method, 1-D KDE for A
(iteration 1), Sample Mean: 285.5412, Sample Std: 105.3427



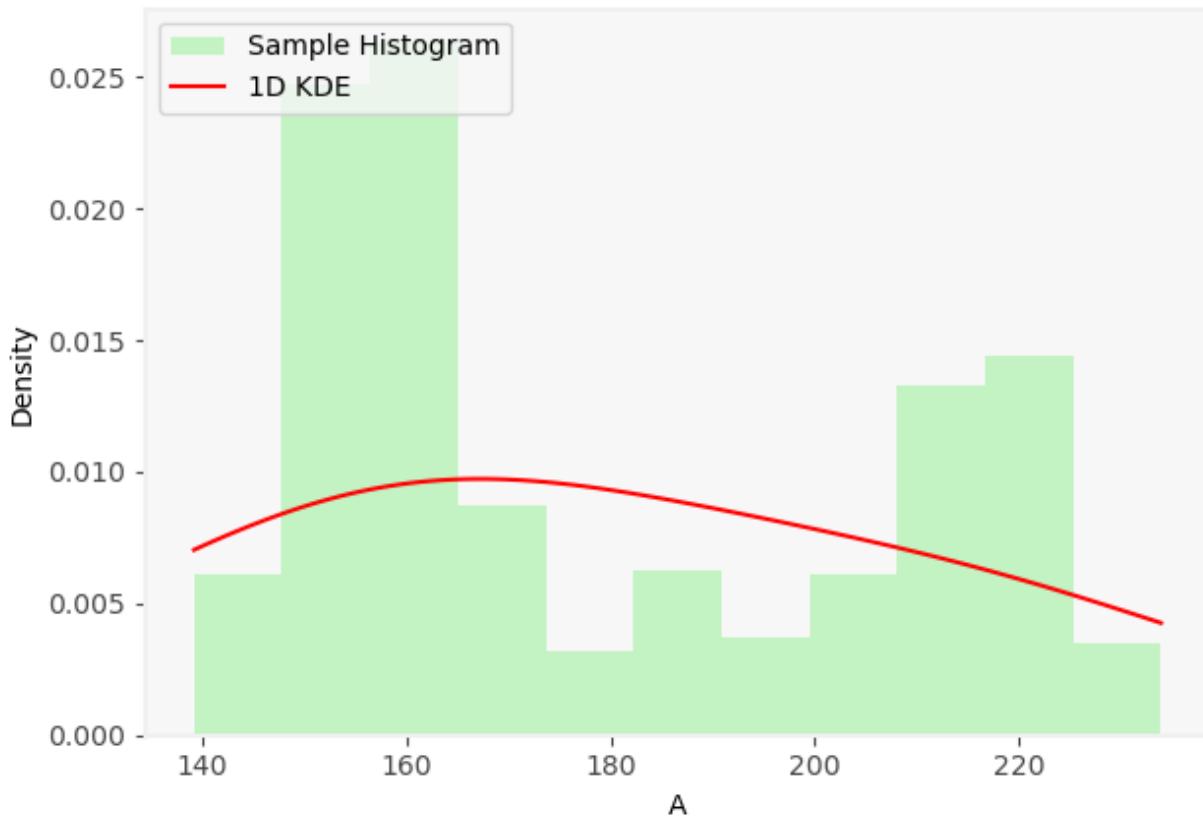
Entropy 1 Method, 1-D KDE for A
(iteration 2), Sample Mean: 188.5481, Sample Std: 41.2680



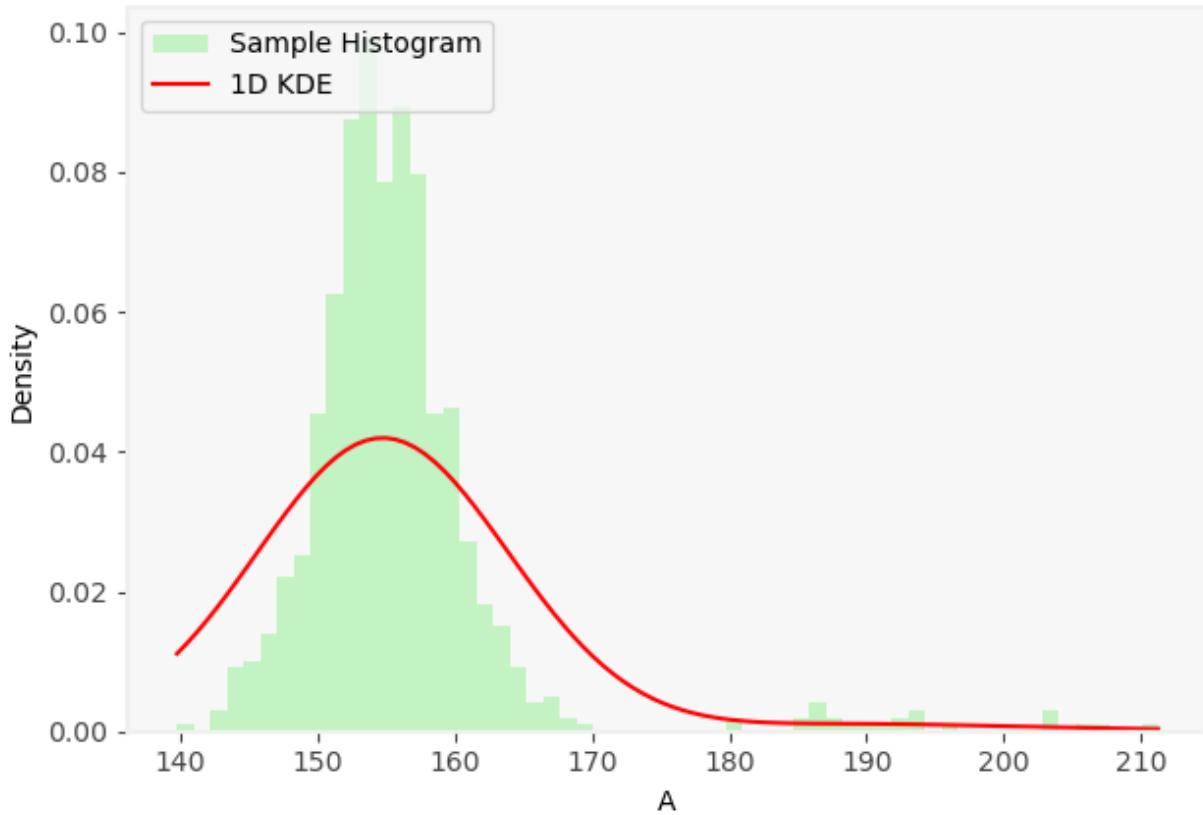
Entropy 1 Method, 1-D KDE for A
(iteration 3), Sample Mean: 203.8177, Sample Std: 26.4381



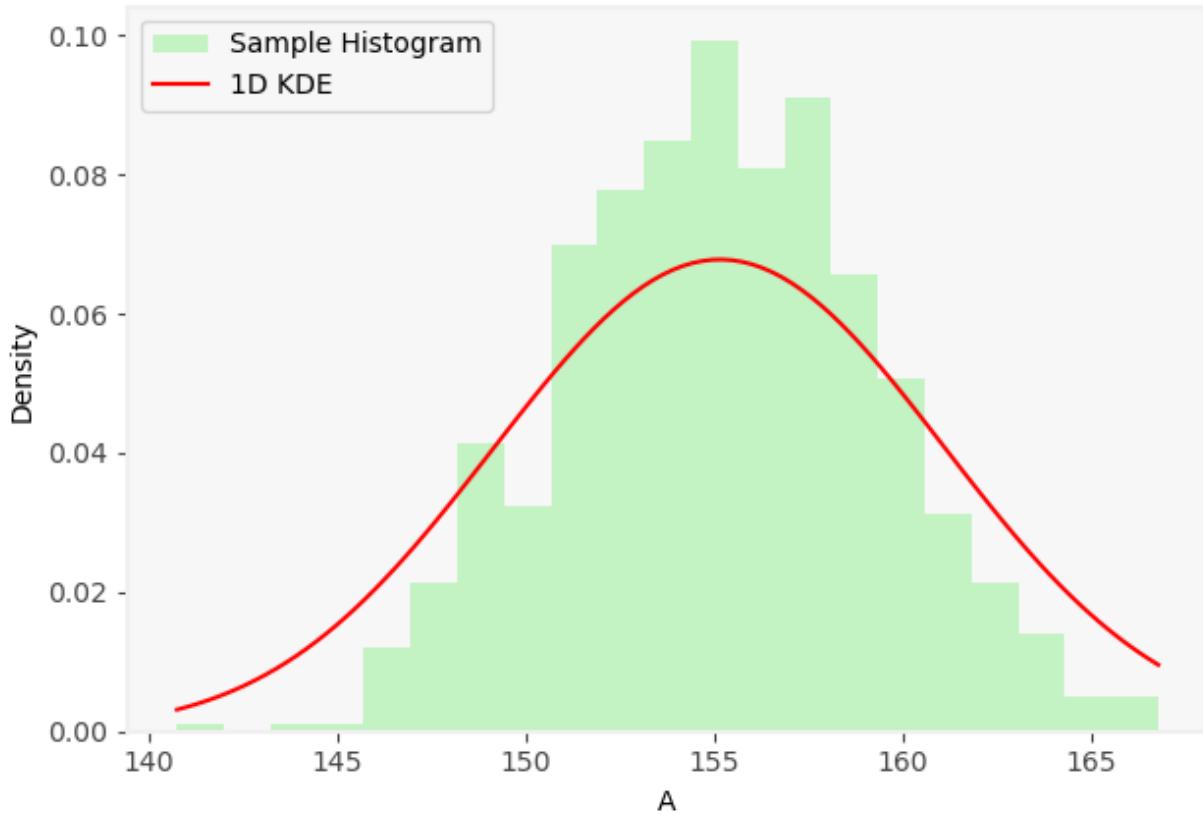
Entropy 1 Method, 1-D KDE for A
(iteration 4), Sample Mean: 179.1089, Sample Std: 27.6092



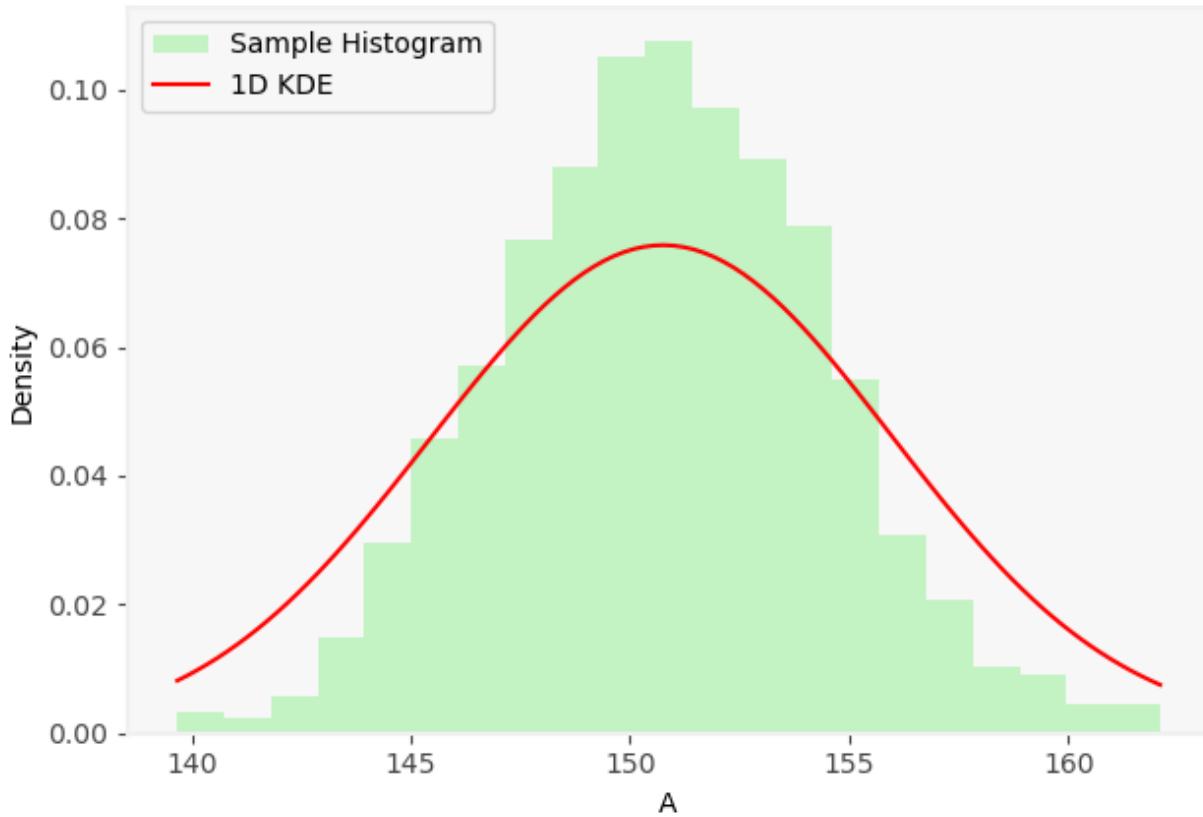
Entropy 1 Method, 1-D KDE for A
(iteration 5), Sample Mean: 155.9281, Sample Std: 8.0585



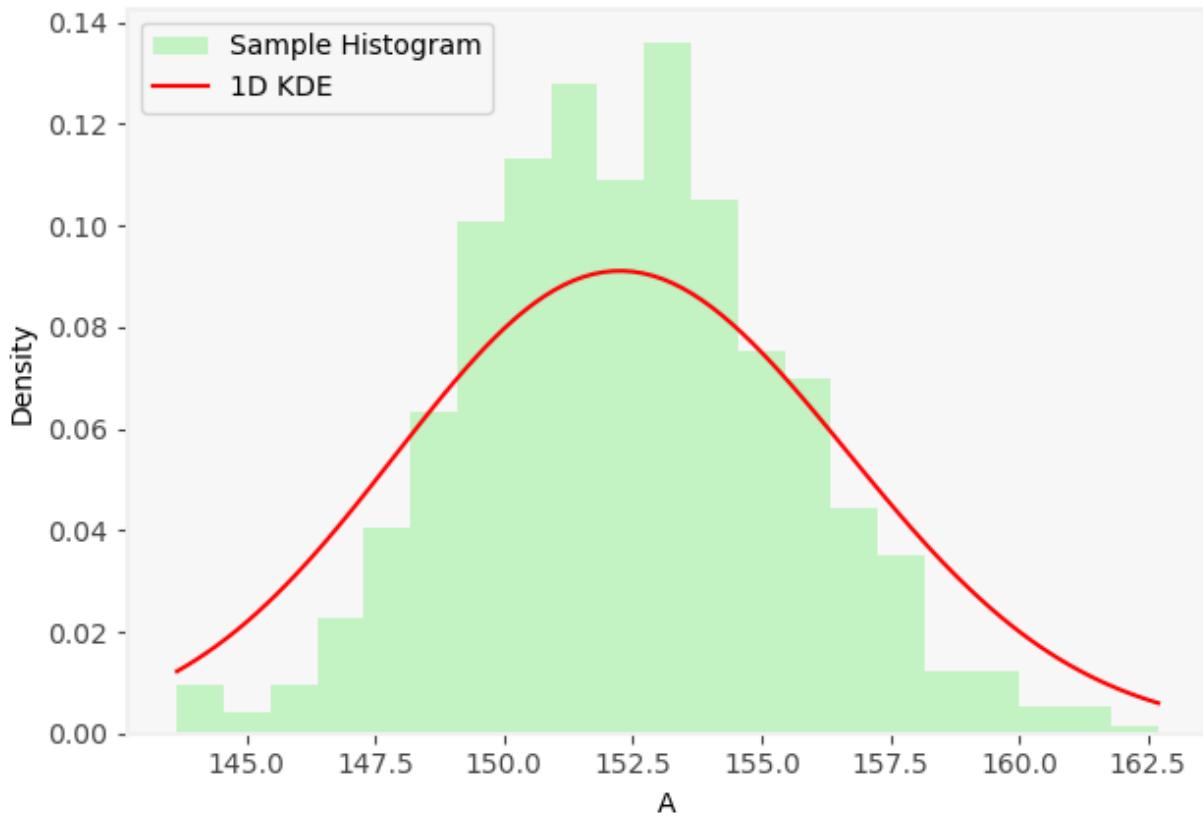
Entropy 1 Method, 1-D KDE for A
(iteration 6), Sample Mean: 155.1580, Sample Std: 4.1243



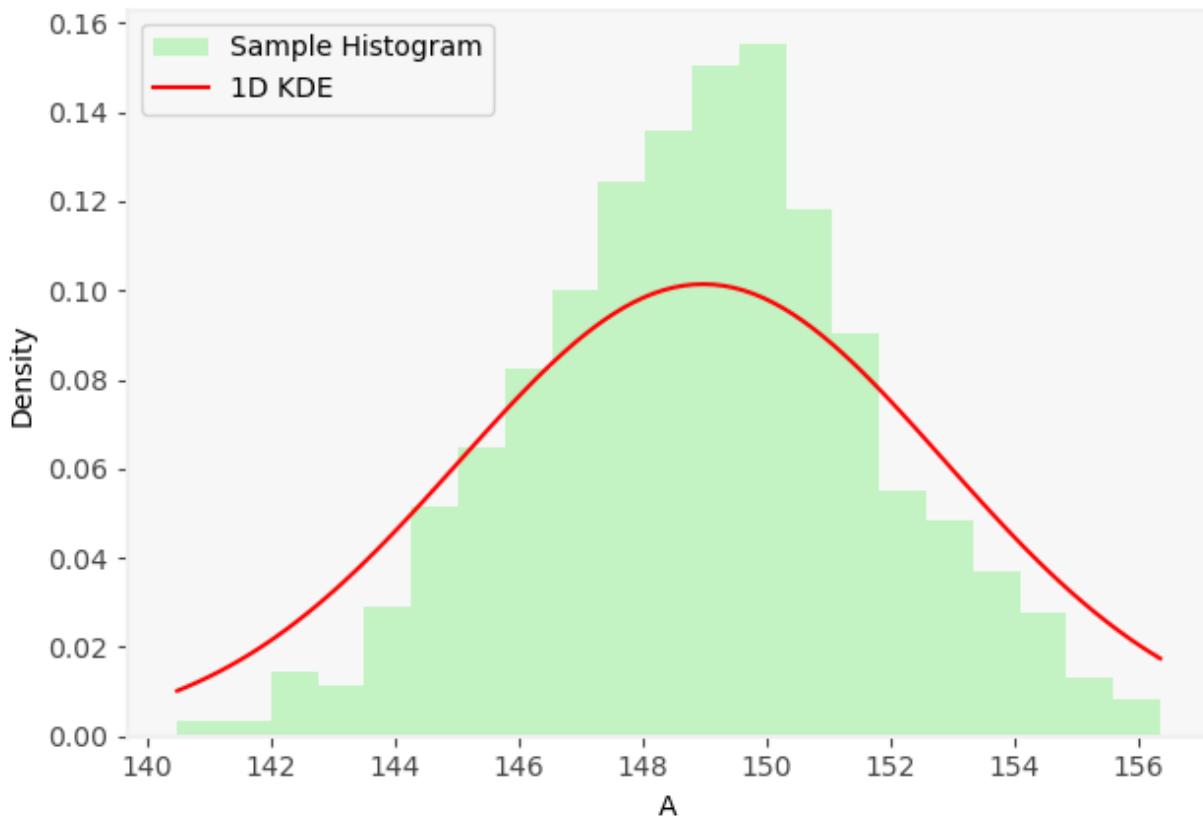
Entropy 1 Method, 1-D KDE for A
(iteration 7), Sample Mean: 150.7597, Sample Std: 3.7162



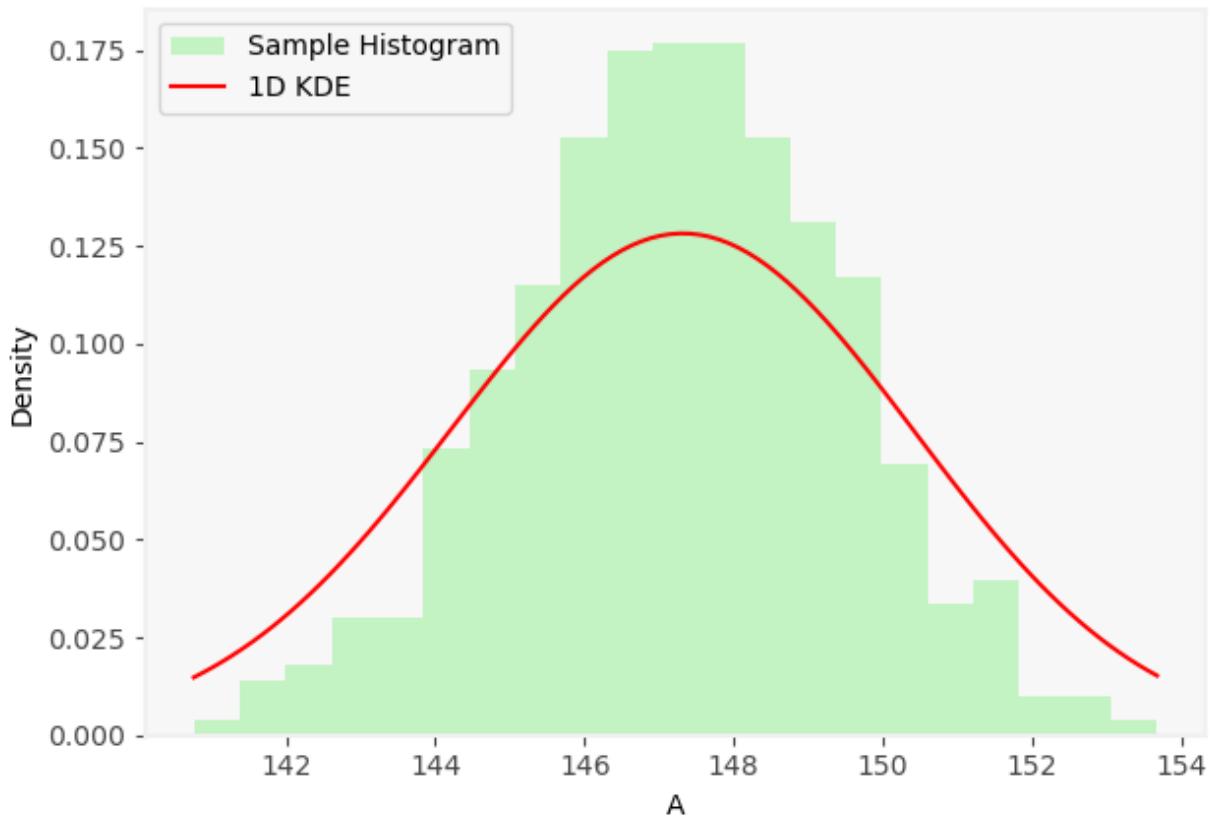
Entropy 1 Method, 1-D KDE for A
(iteration 8), Sample Mean: 152.3748, Sample Std: 3.0992



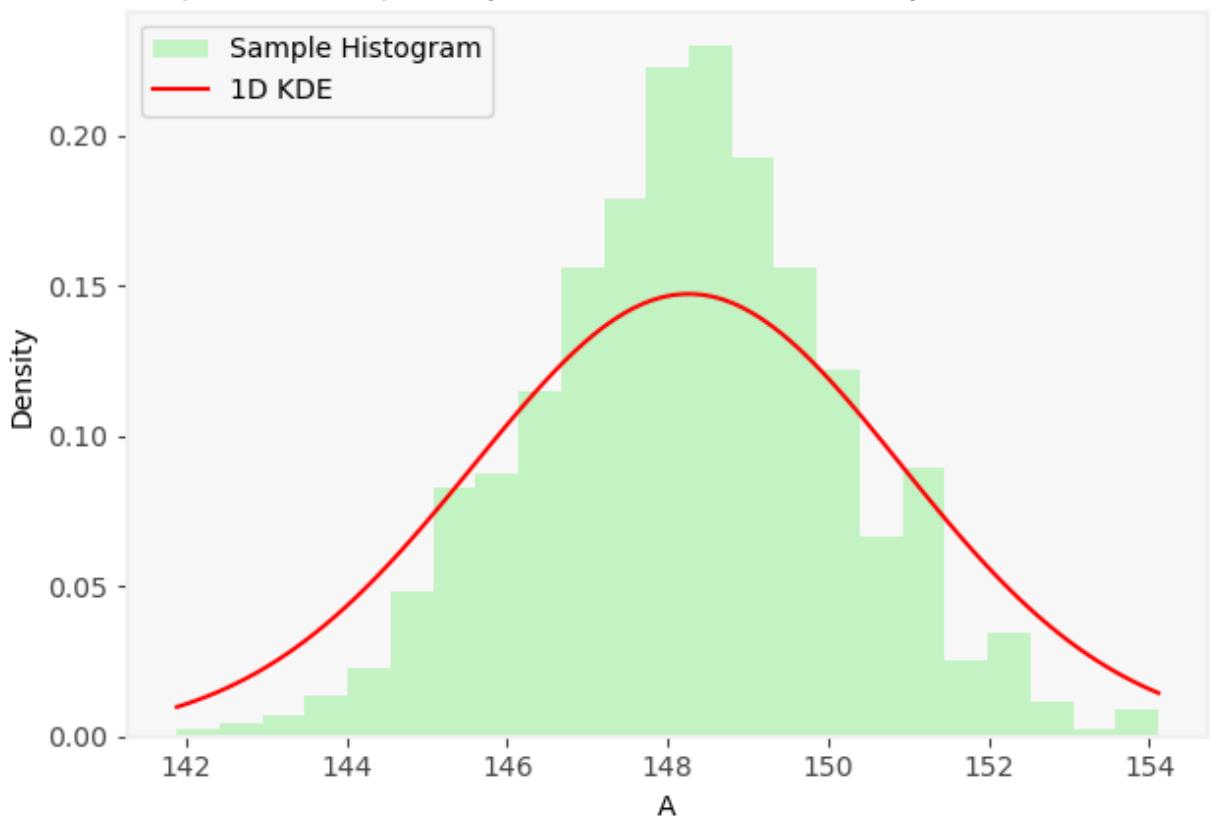
Entropy 1 Method, 1-D KDE for A
(iteration 9), Sample Mean: 148.9554, Sample Std: 2.7784



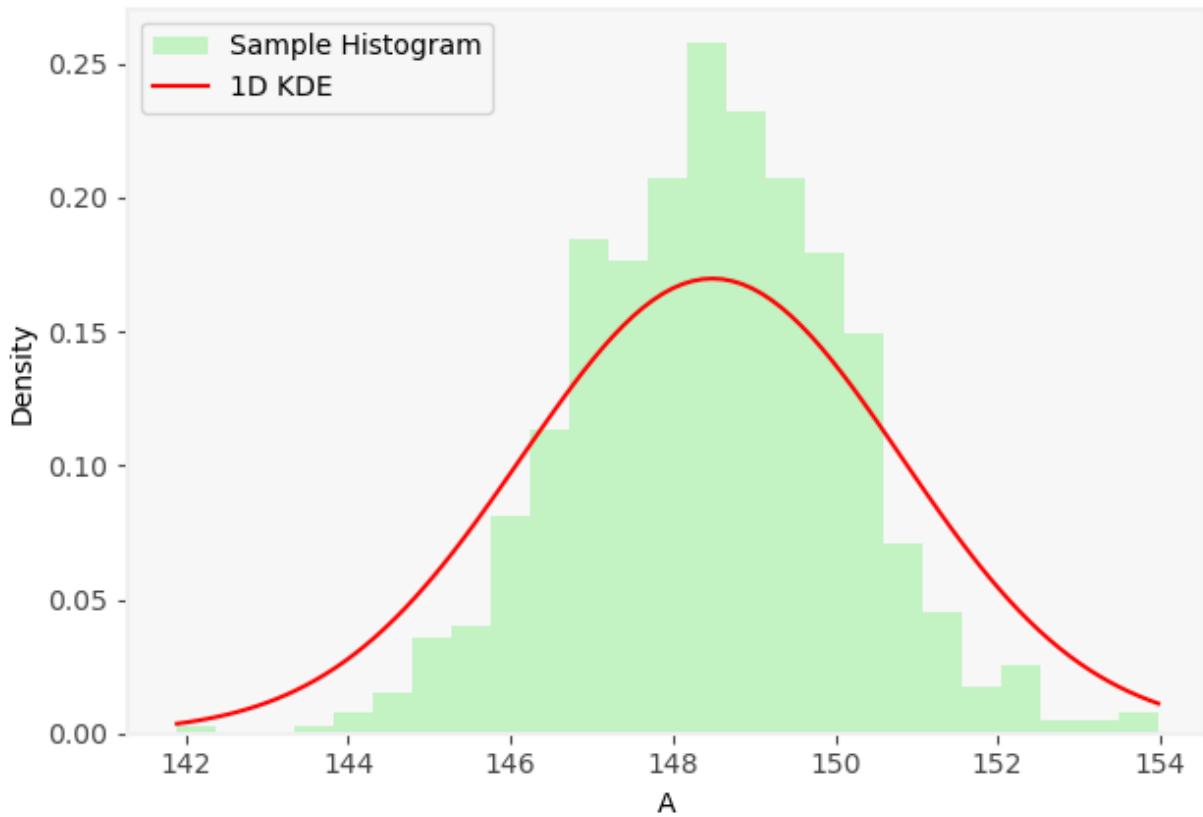
Entropy 1 Method, 1-D KDE for A
(iteration 10), Sample Mean: 147.2701, Sample Std: 2.1978



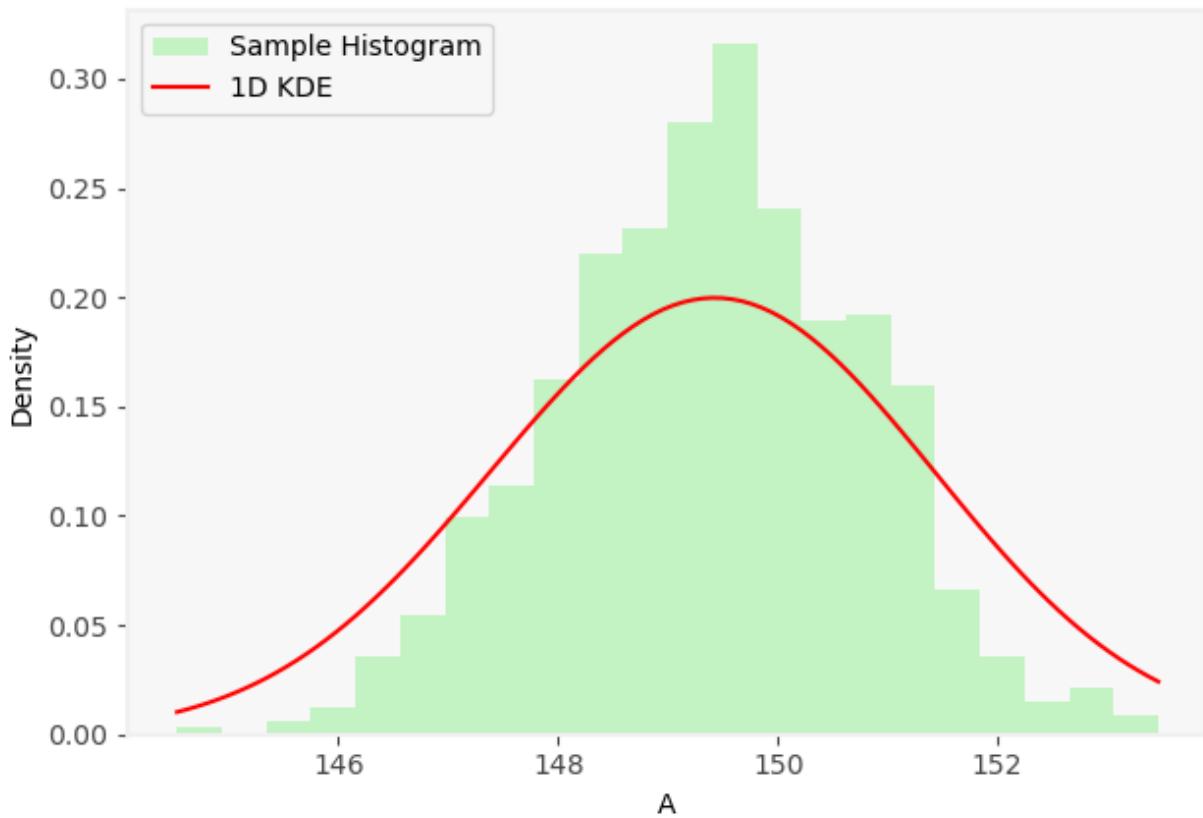
Entropy 1 Method, 1-D KDE for A
(iteration 11), Sample Mean: 148.2449, Sample Std: 1.9266

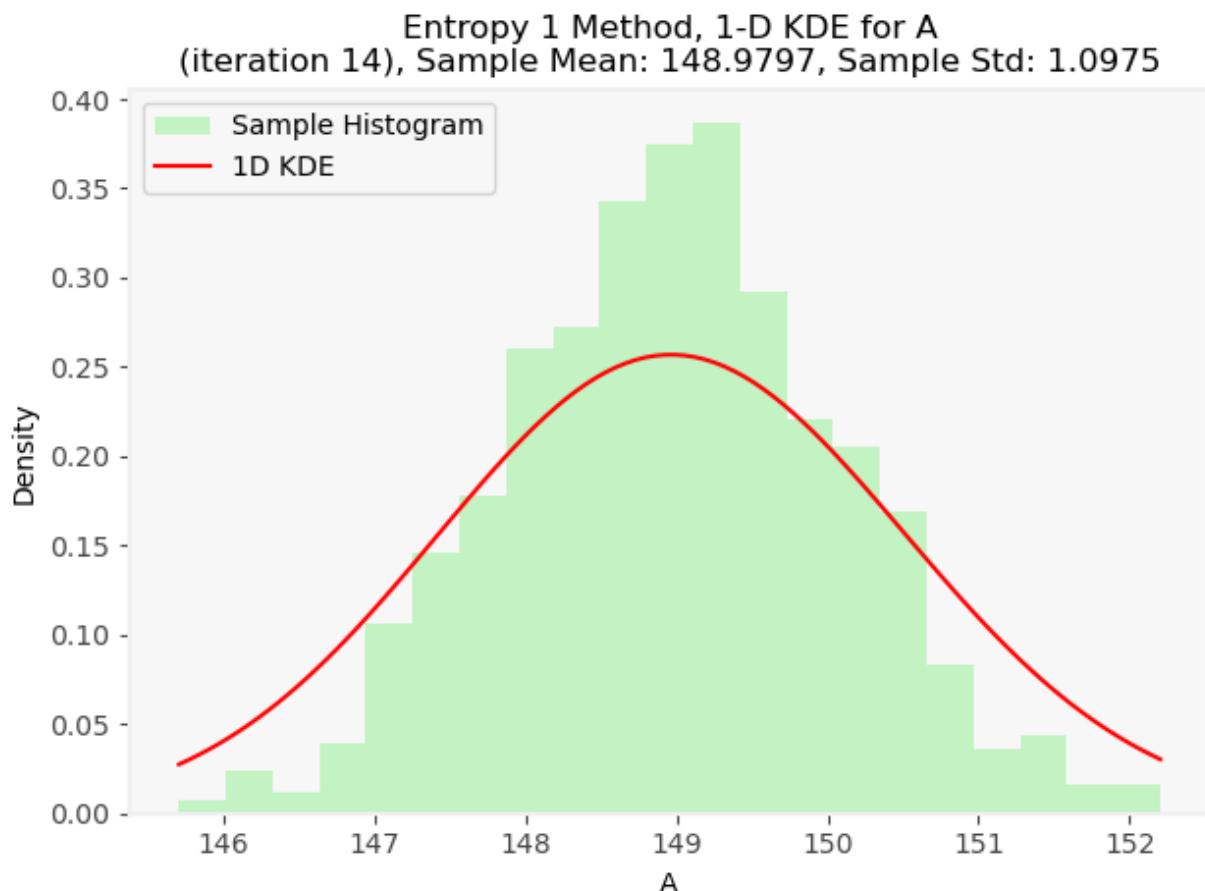


Entropy 1 Method, 1-D KDE for A
(iteration 12), Sample Mean: 148.4715, Sample Std: 1.6720



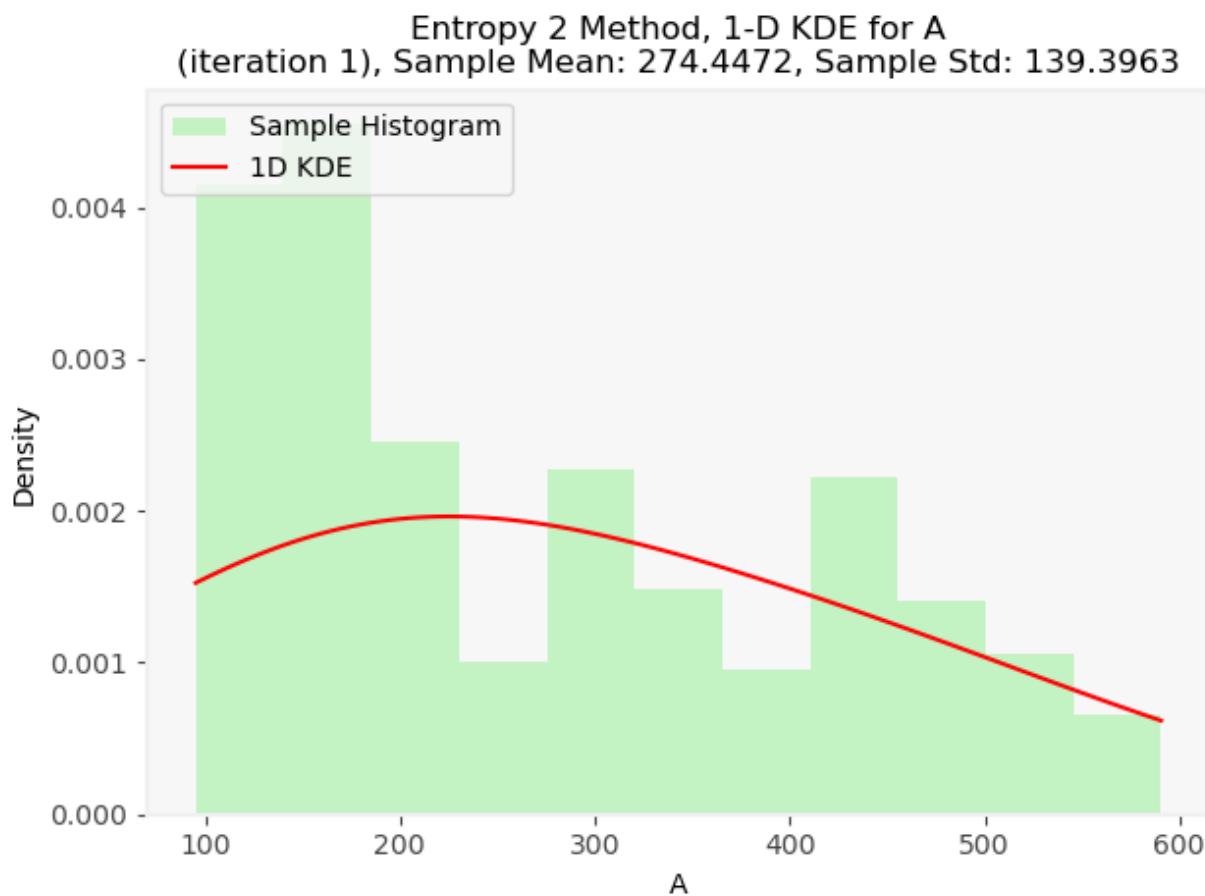
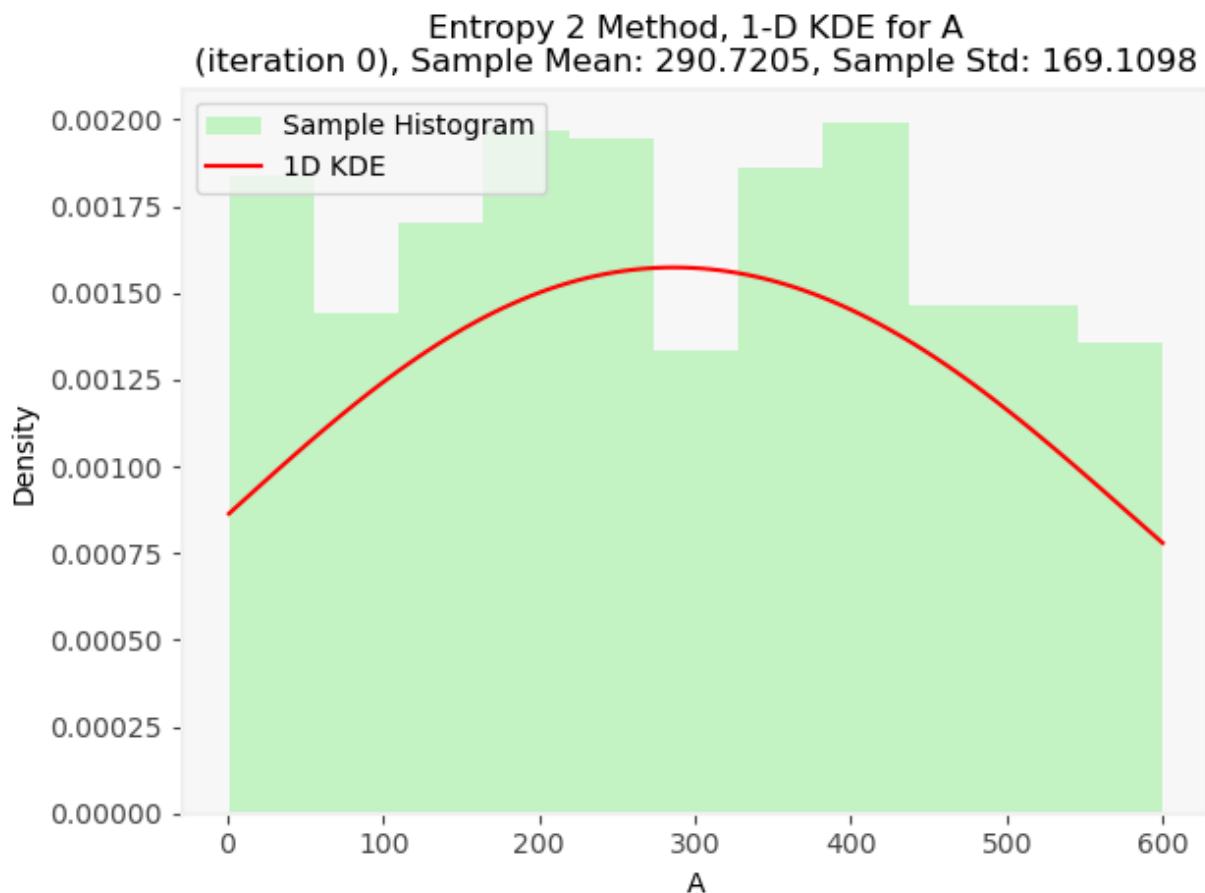
Entropy 1 Method, 1-D KDE for A
(iteration 13), Sample Mean: 149.3860, Sample Std: 1.4029



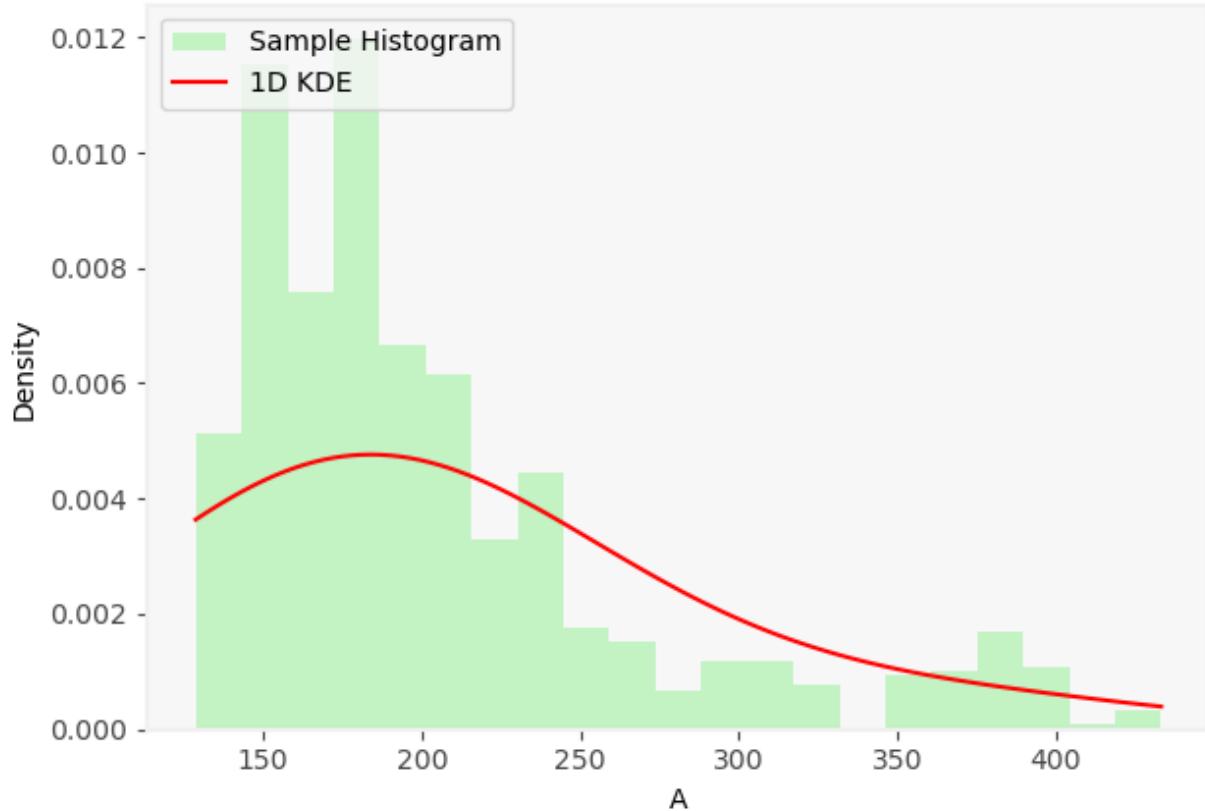


In []:

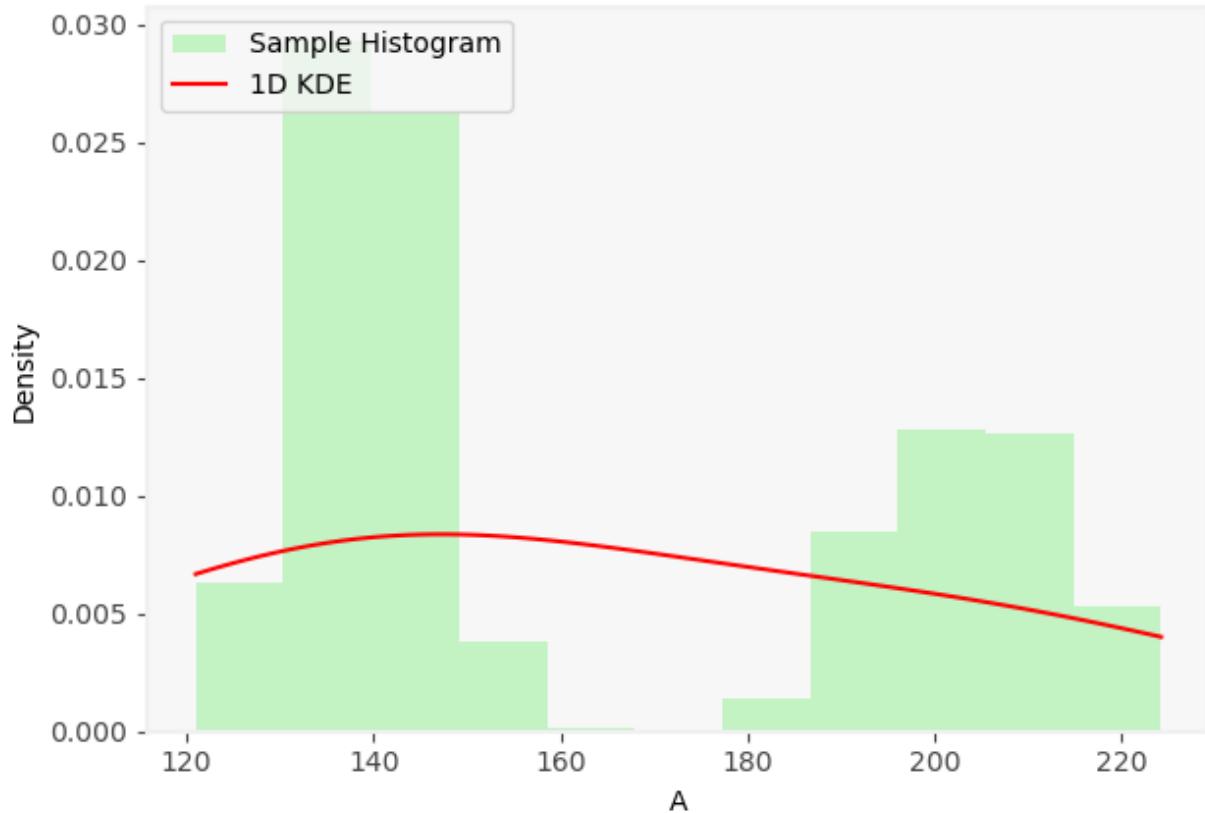
```
In [53]: for i in range(len(exp_entropy2.totaltimes())):
    MyPlots.plot_hist_1d_kde(list_par_separated_e2[0][i], kdes_entropy2[i,0],"Entropy")
```



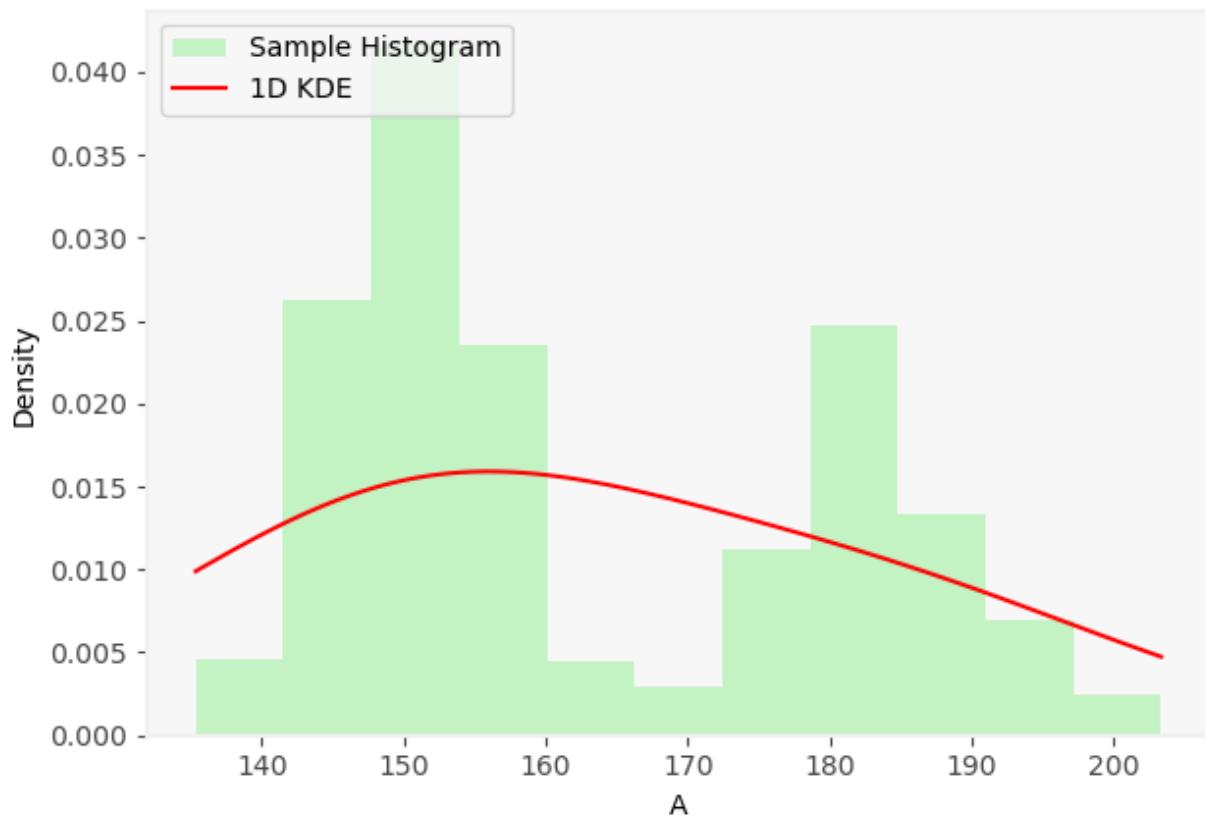
Entropy 2 Method, 1-D KDE for A
(iteration 2), Sample Mean: 205.3310, Sample Std: 65.4938



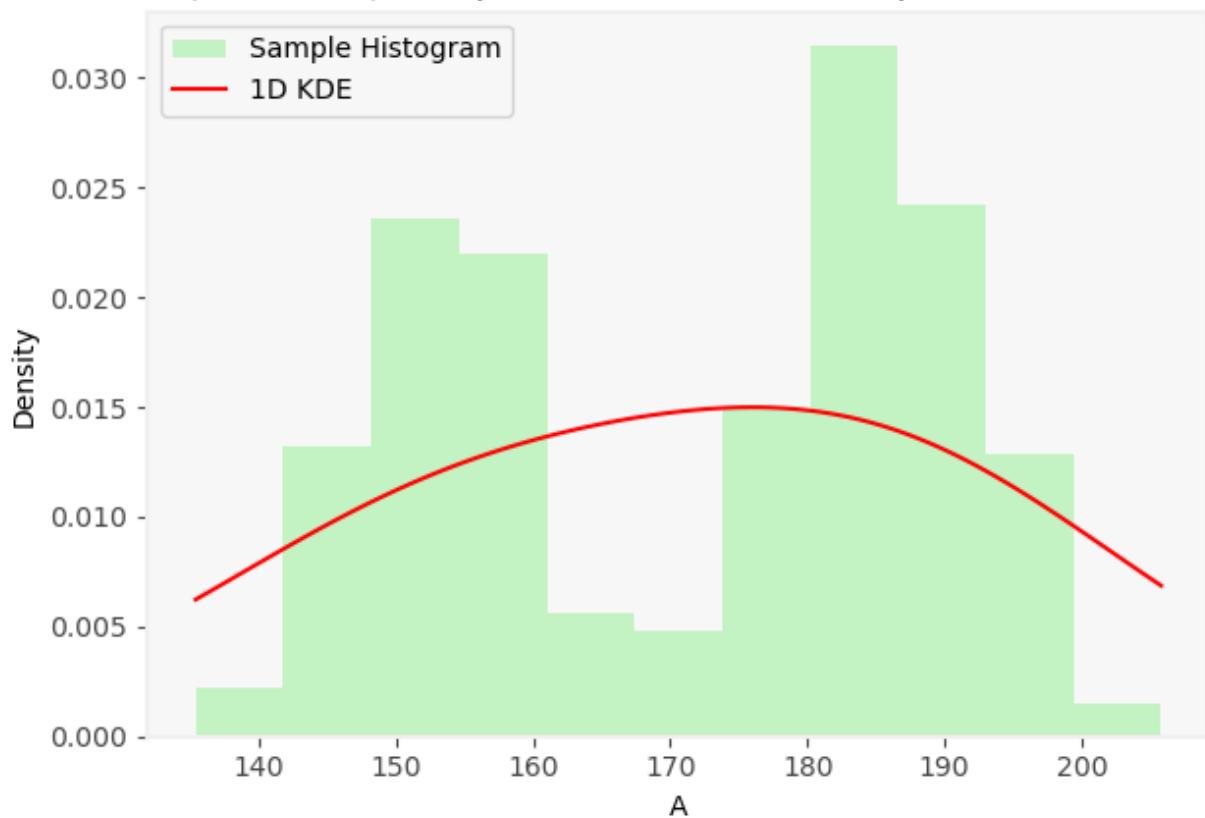
Entropy 2 Method, 1-D KDE for A
(iteration 3), Sample Mean: 163.5642, Sample Std: 32.3217



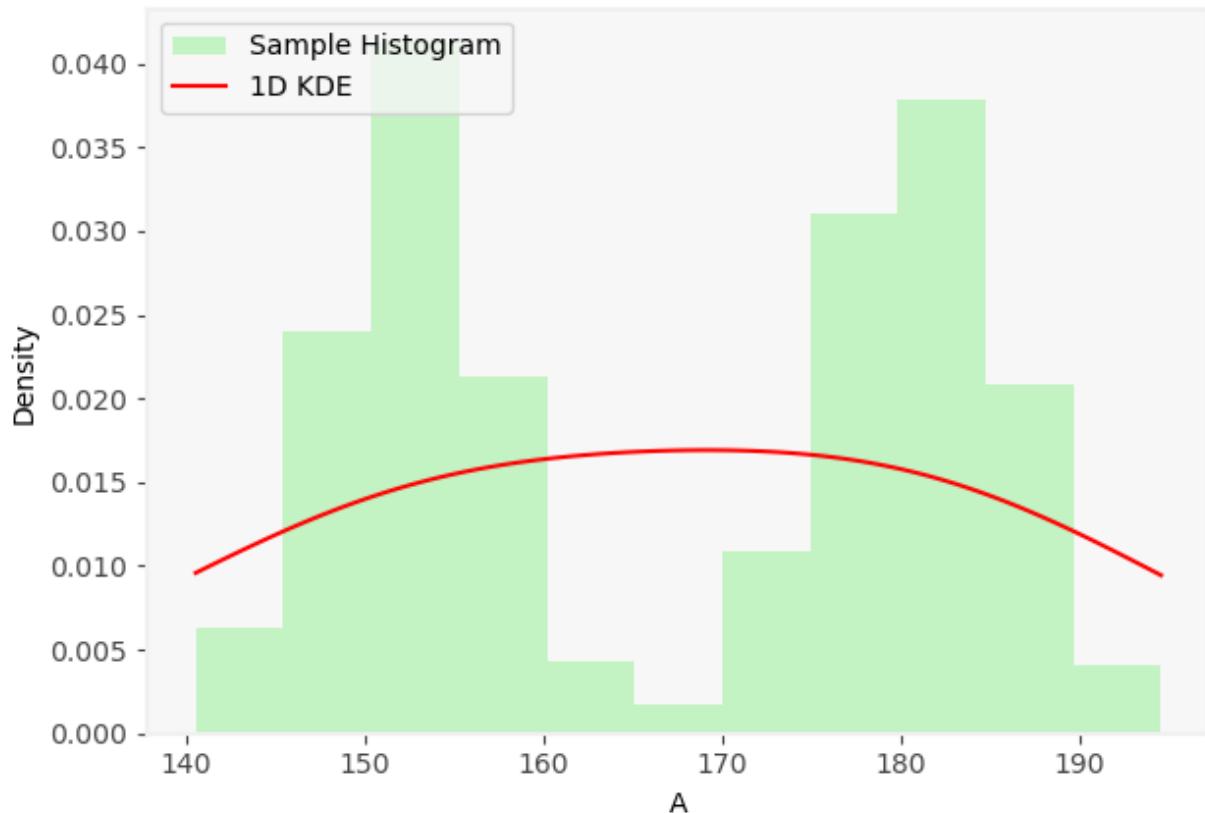
Entropy 2 Method, 1-D KDE for A
(iteration 4), Sample Mean: 163.0431, Sample Std: 17.0906



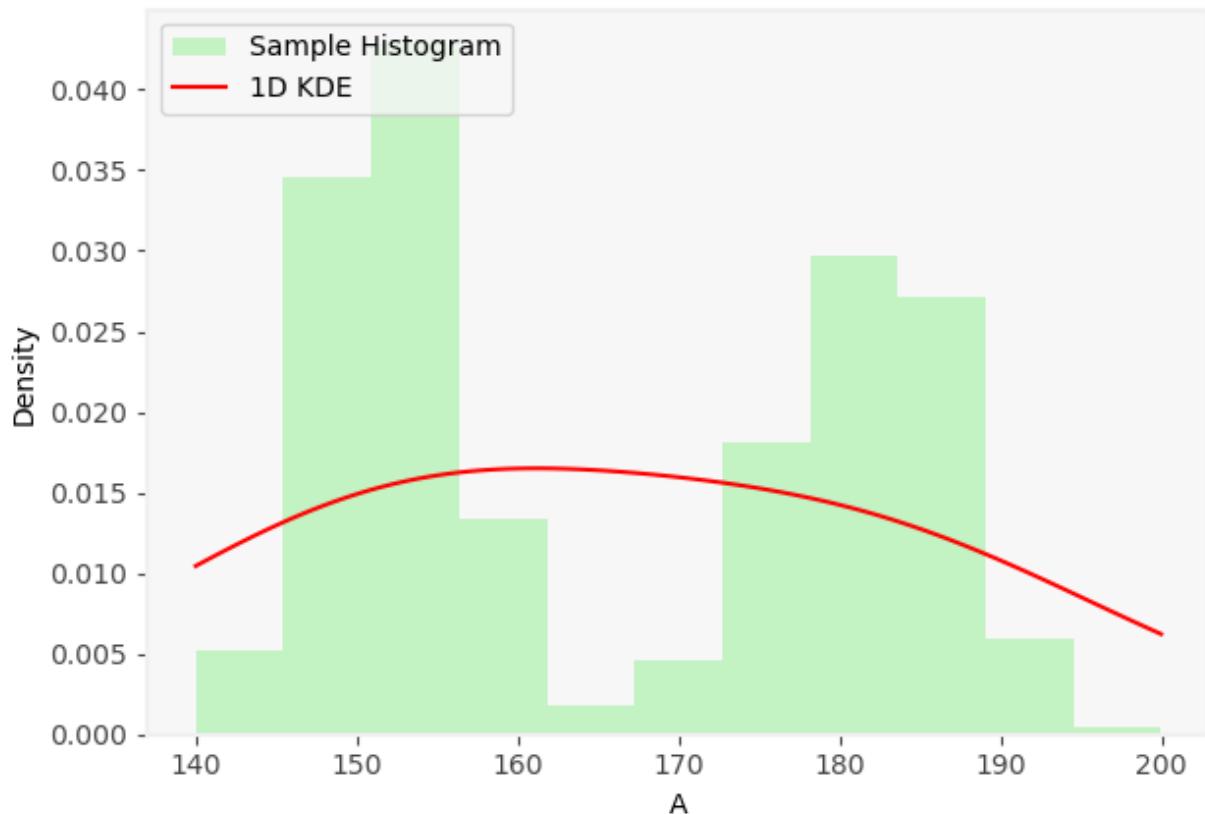
Entropy 2 Method, 1-D KDE for A
(iteration 5), Sample Mean: 171.4617, Sample Std: 17.3457



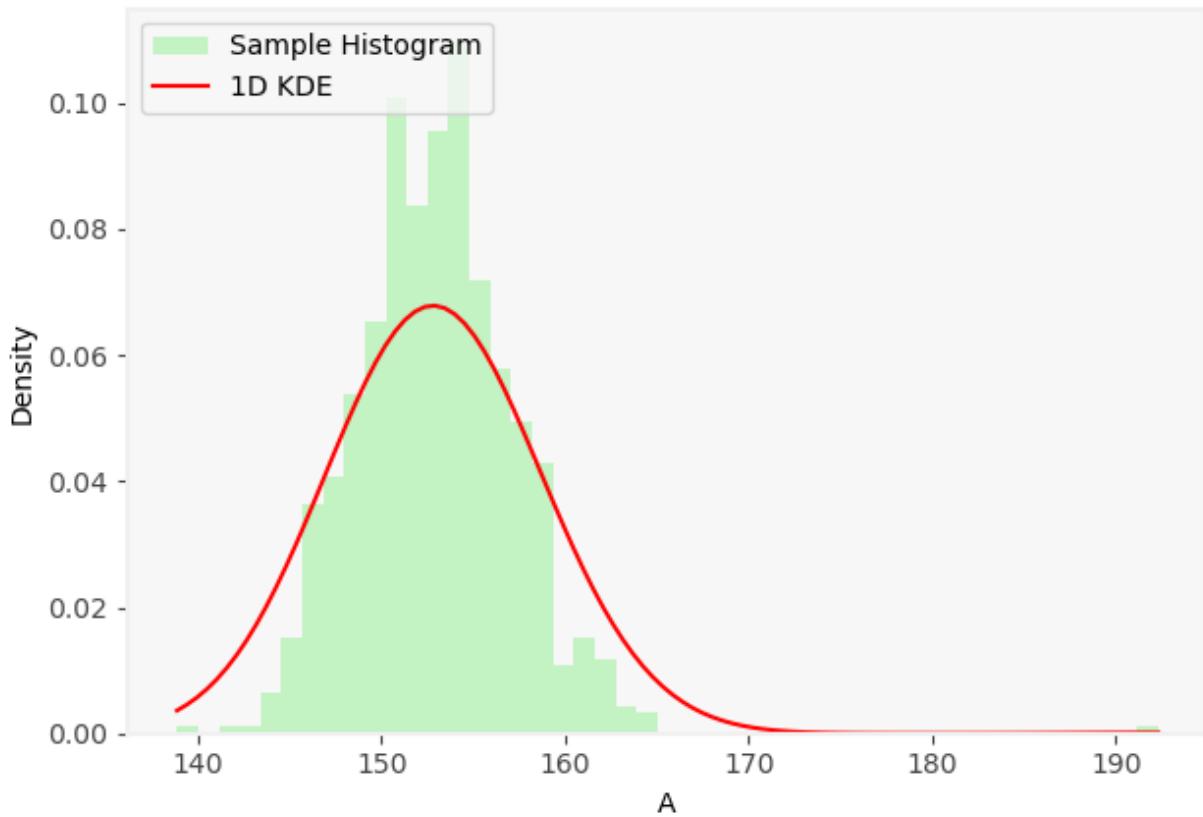
Entropy 2 Method, 1-D KDE for A
(iteration 6), Sample Mean: 167.3749, Sample Std: 14.9495



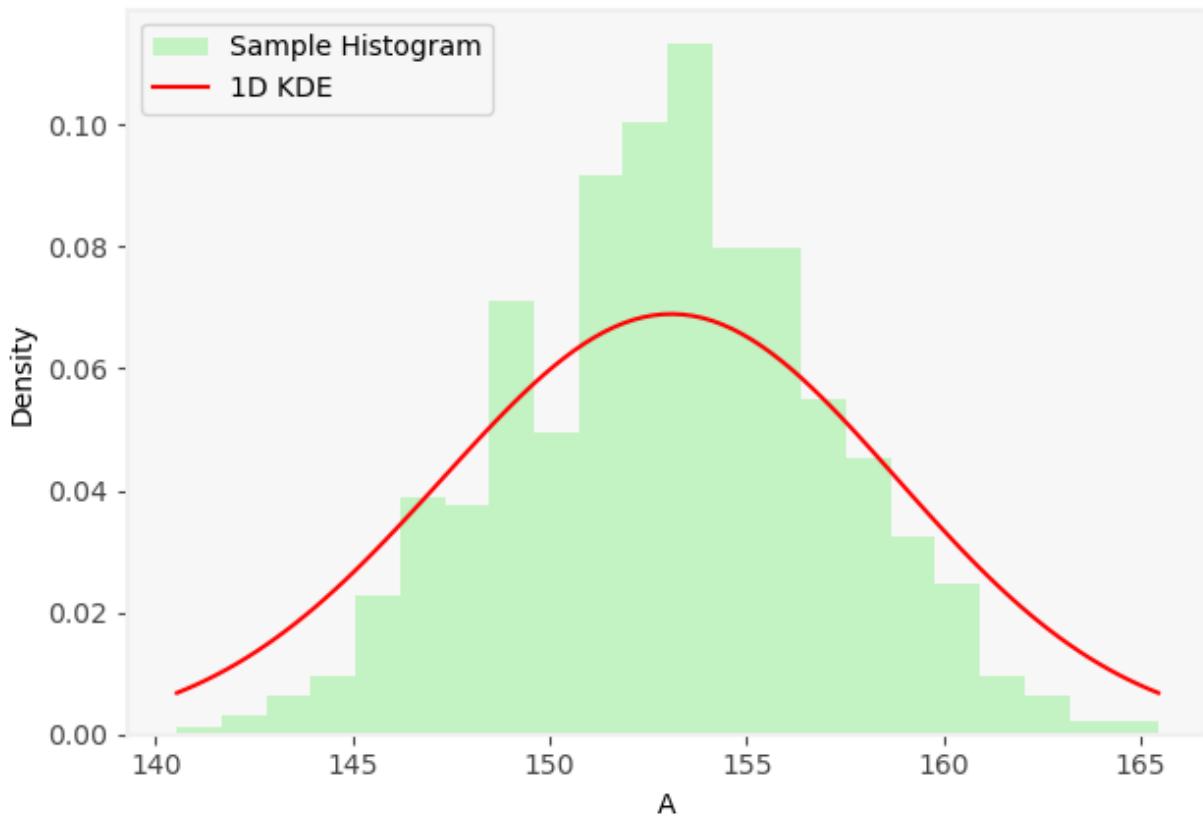
Entropy 2 Method, 1-D KDE for A
(iteration 7), Sample Mean: 165.8514, Sample Std: 15.4860



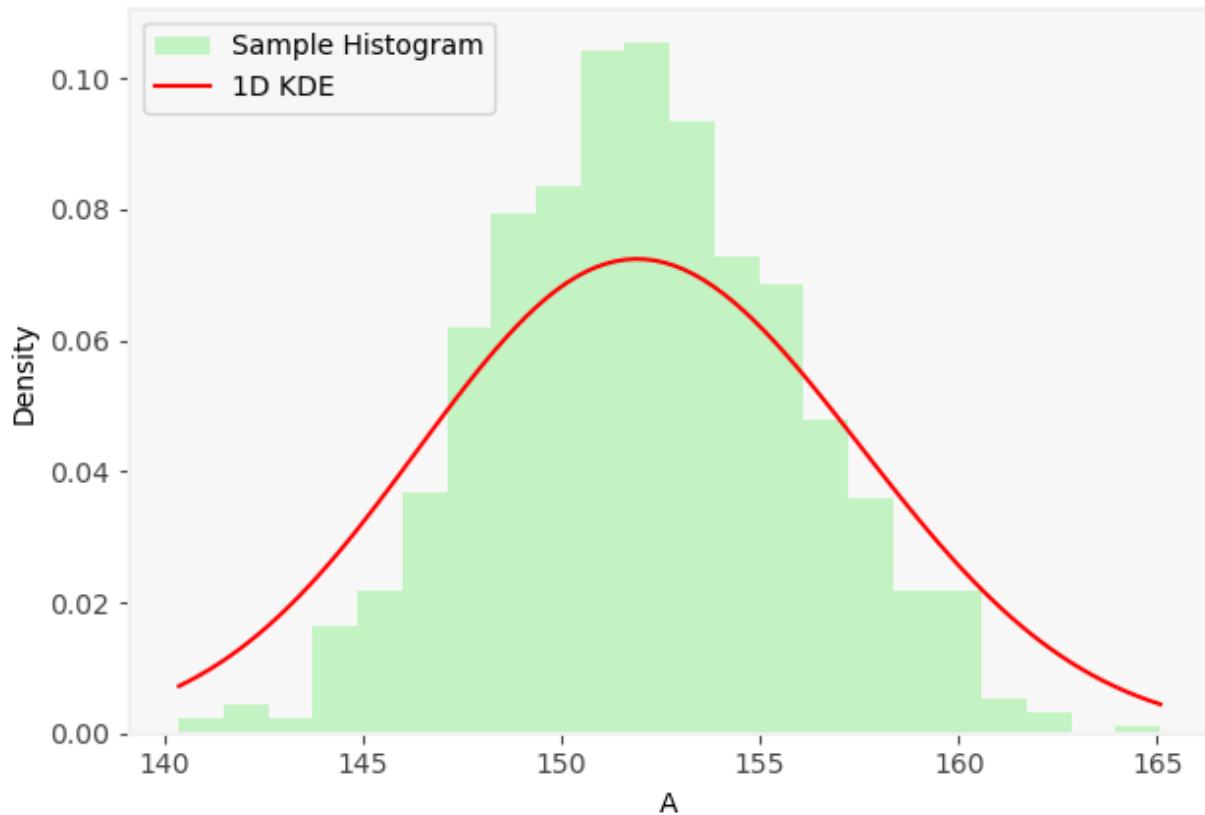
Entropy 2 Method, 1-D KDE for A
(iteration 8), Sample Mean: 152.9567, Sample Std: 4.2533



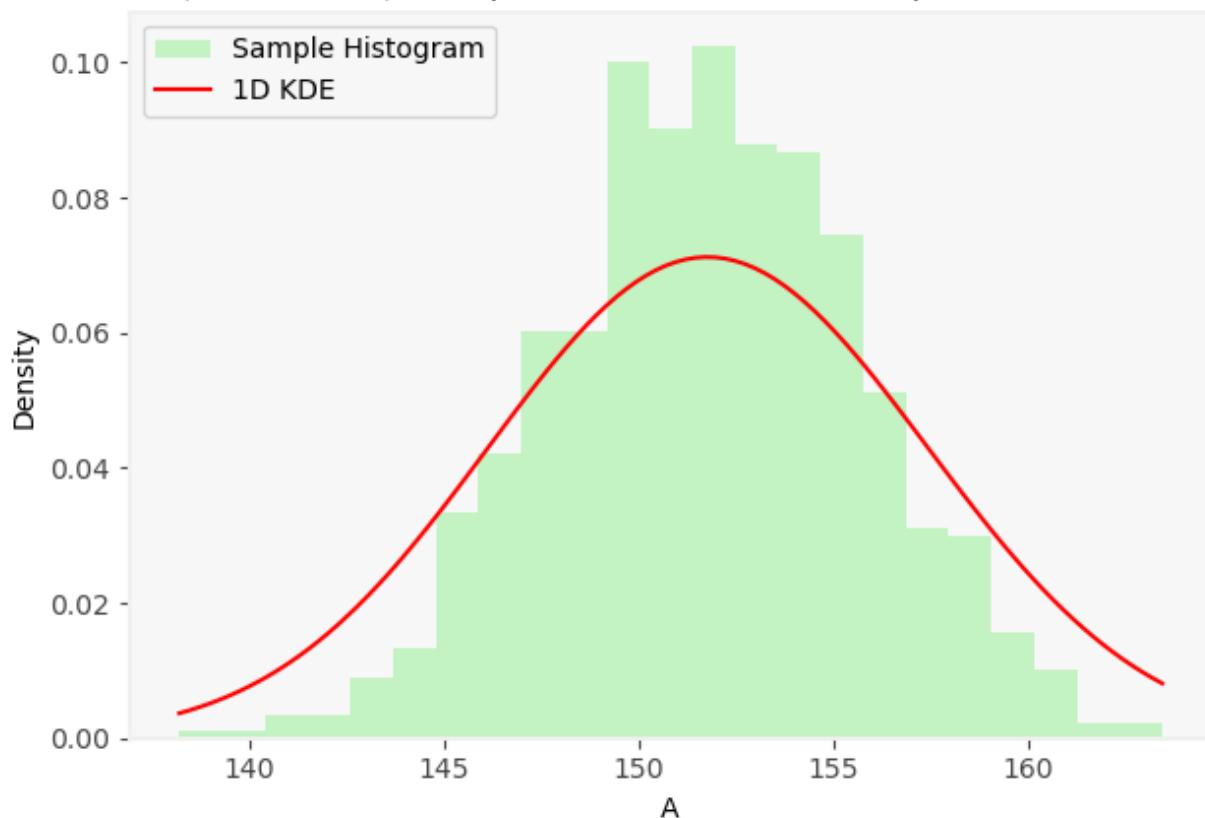
Entropy 2 Method, 1-D KDE for A
(iteration 9), Sample Mean: 153.0261, Sample Std: 4.0806



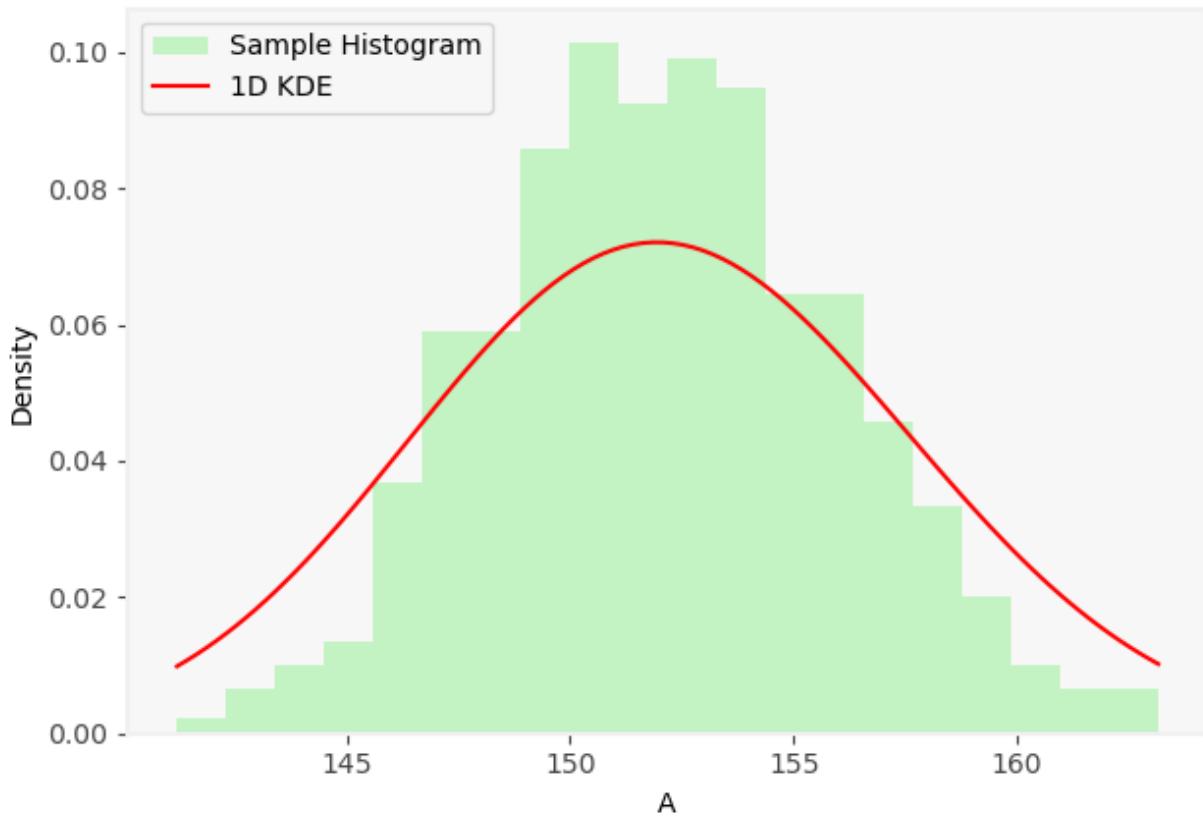
Entropy 2 Method, 1-D KDE for A
(iteration 10), Sample Mean: 152.0588, Sample Std: 3.8703



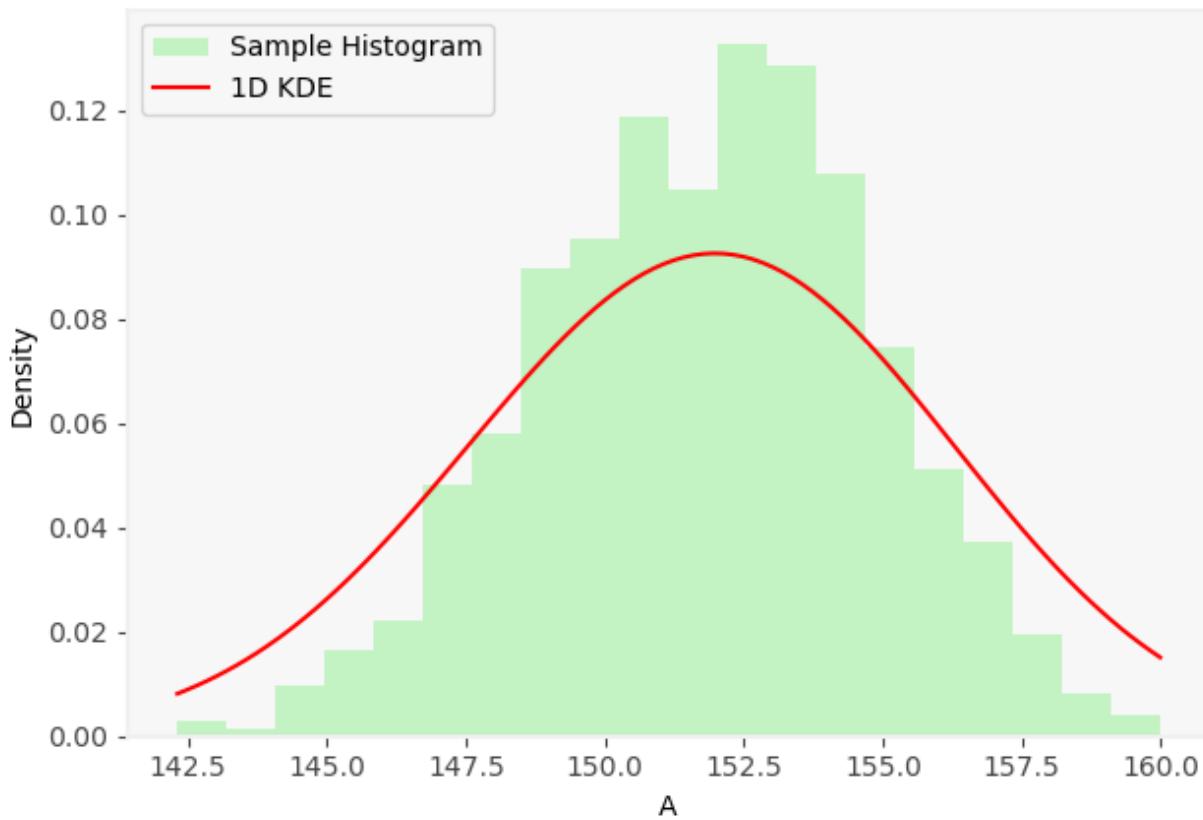
Entropy 2 Method, 1-D KDE for A
(iteration 11), Sample Mean: 151.7600, Sample Std: 3.9437



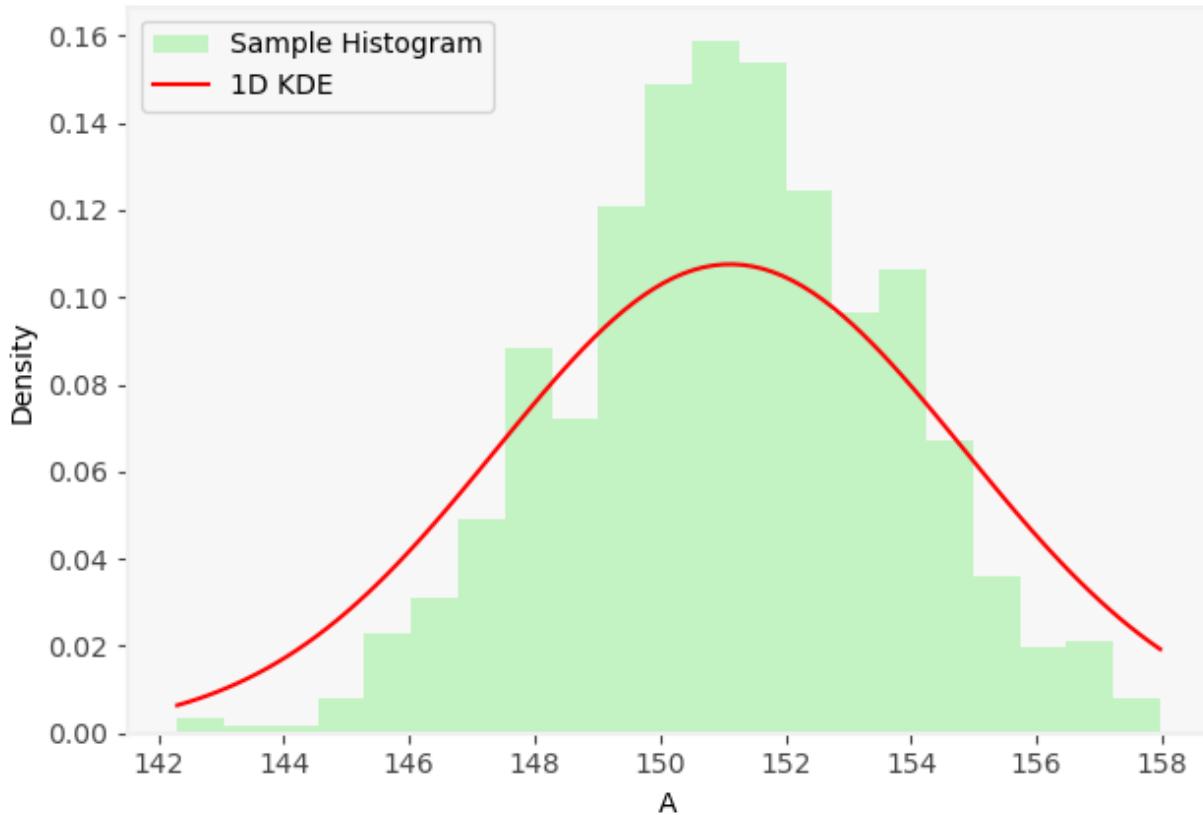
Entropy 2 Method, 1-D KDE for A
(iteration 12), Sample Mean: 152.1472, Sample Std: 3.8853



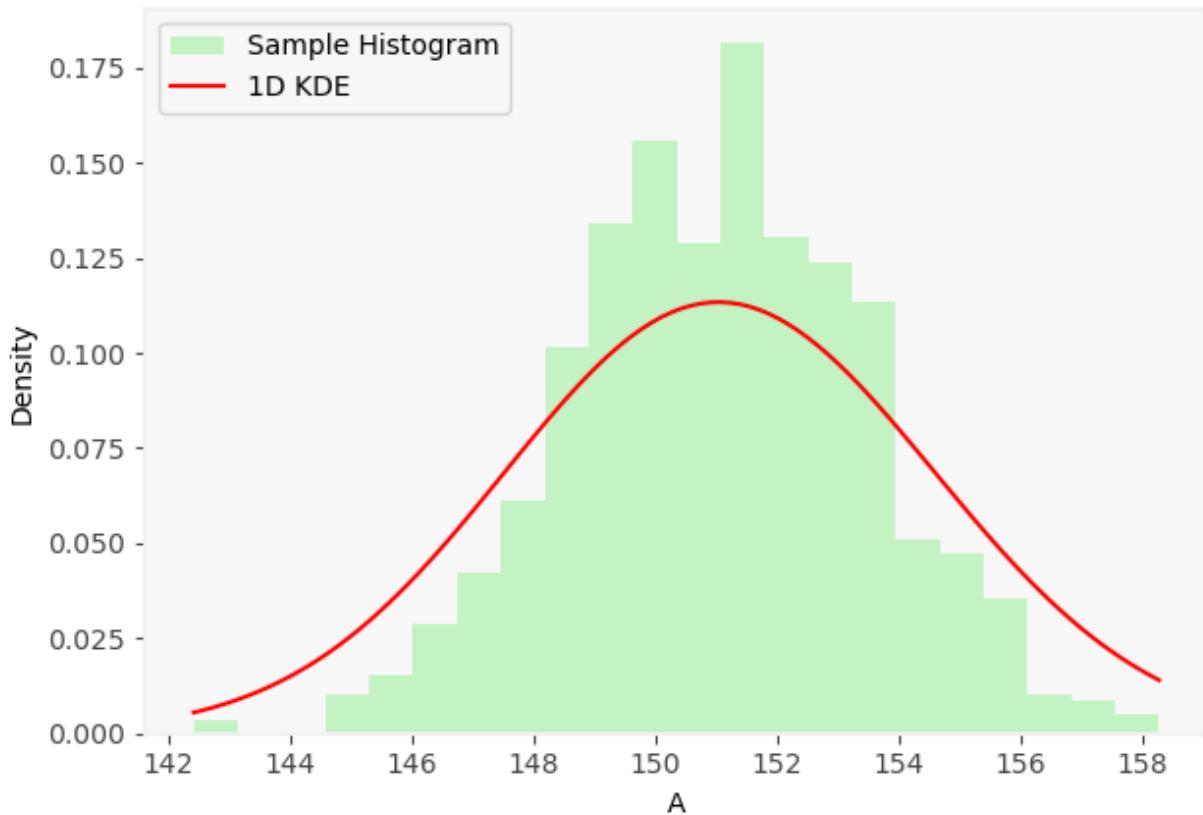
Entropy 2 Method, 1-D KDE for A
(iteration 13), Sample Mean: 151.8370, Sample Std: 3.0244



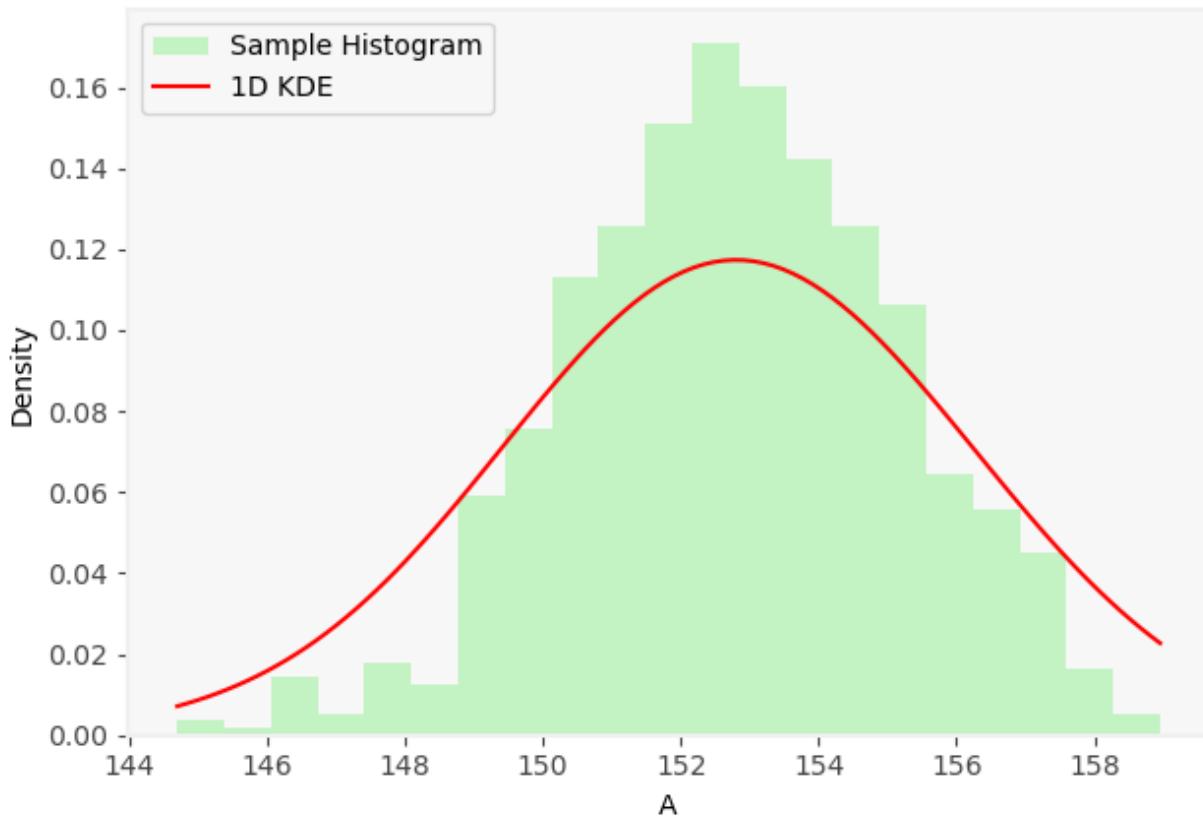
Entropy 2 Method, 1-D KDE for A
(iteration 14), Sample Mean: 151.0931, Sample Std: 2.6100



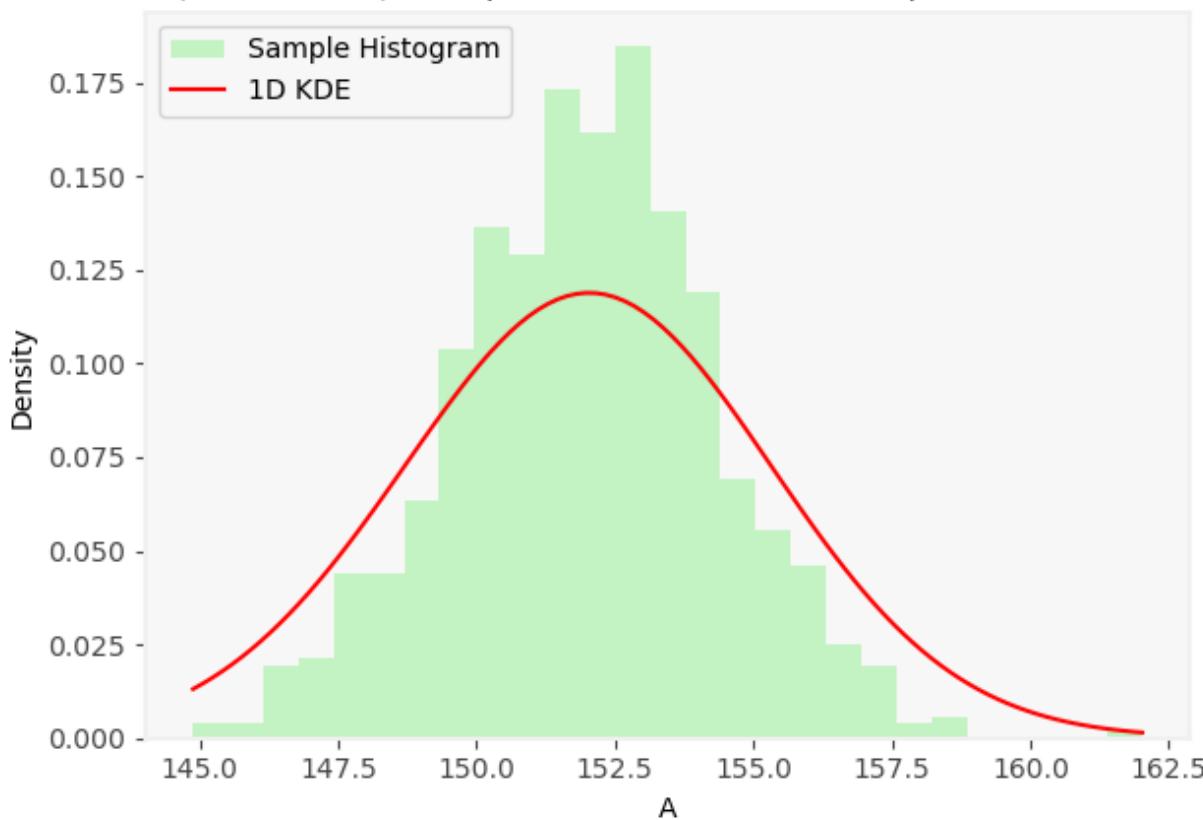
Entropy 2 Method, 1-D KDE for A
(iteration 15), Sample Mean: 151.0566, Sample Std: 2.4836



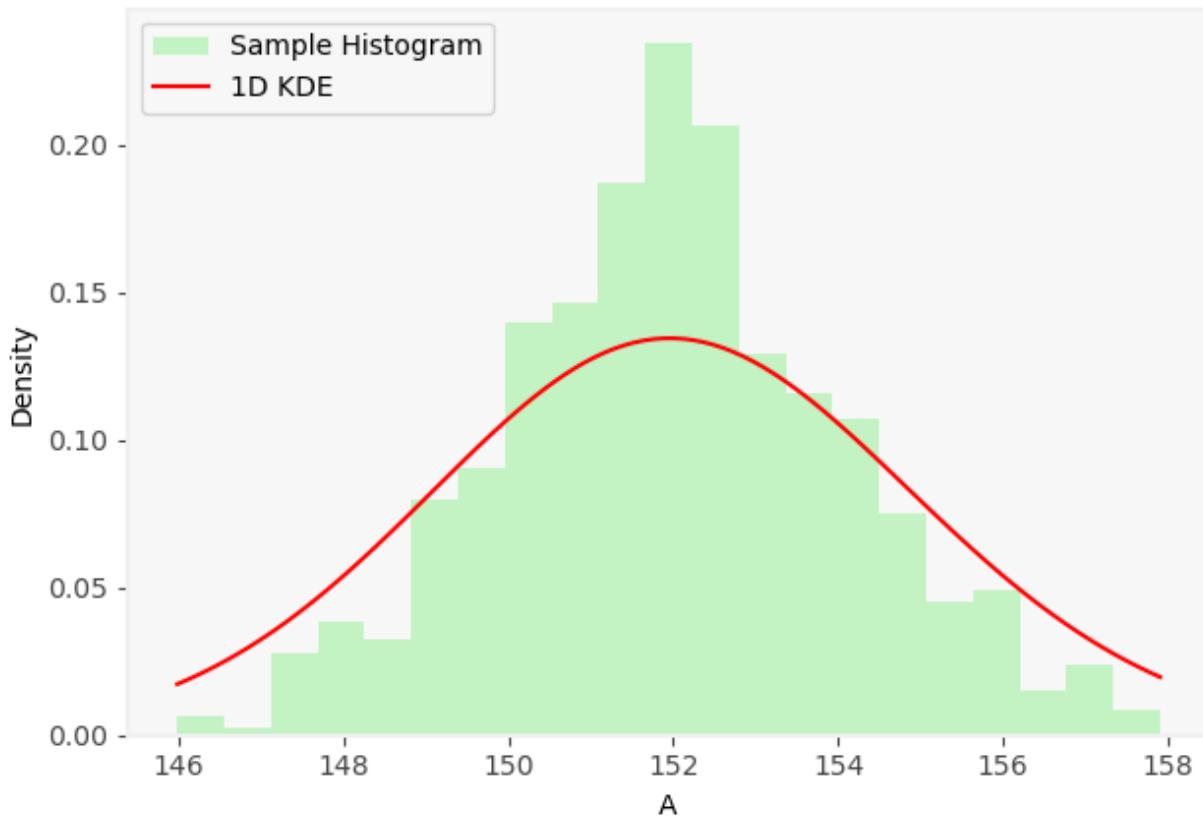
Entropy 2 Method, 1-D KDE for A
(iteration 16), Sample Mean: 152.7794, Sample Std: 2.3972



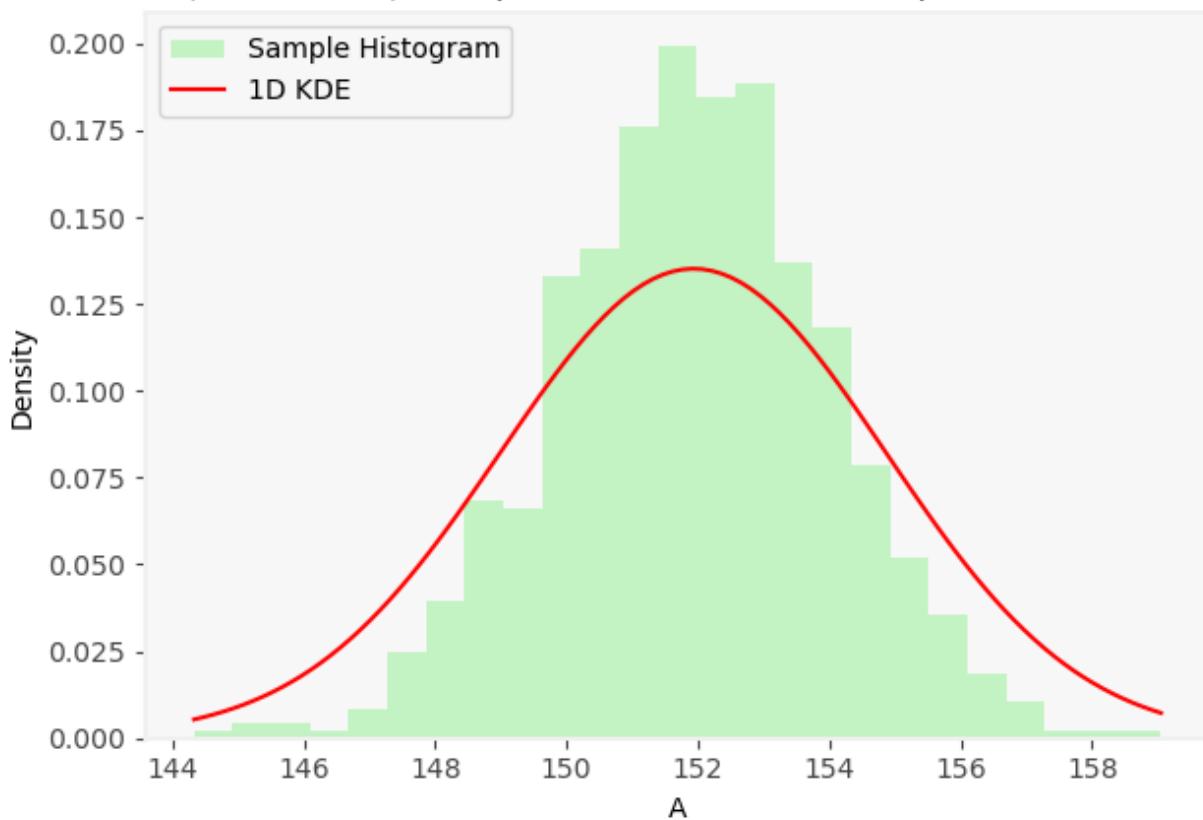
Entropy 2 Method, 1-D KDE for A
(iteration 17), Sample Mean: 151.9937, Sample Std: 2.3906



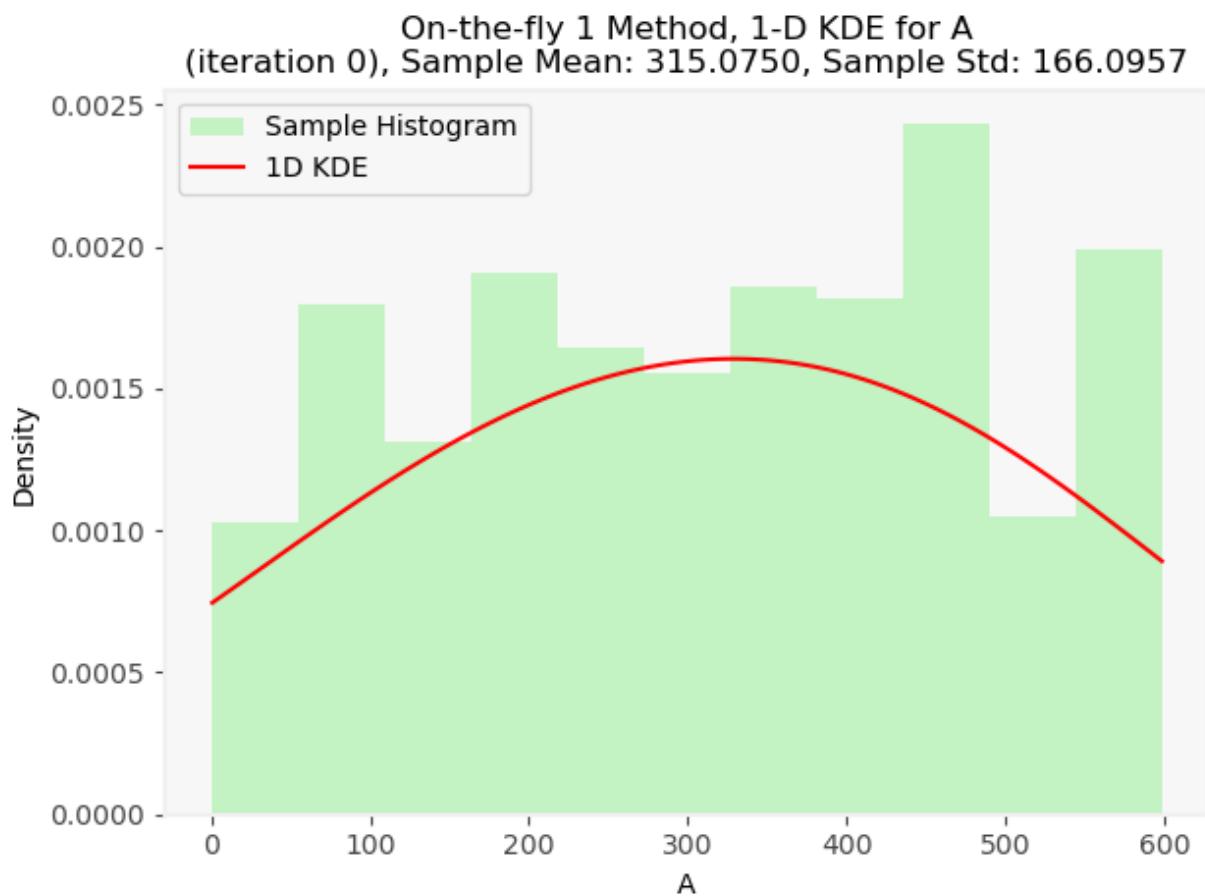
Entropy 2 Method, 1-D KDE for A
(iteration 18), Sample Mean: 152.0167, Sample Std: 2.1131

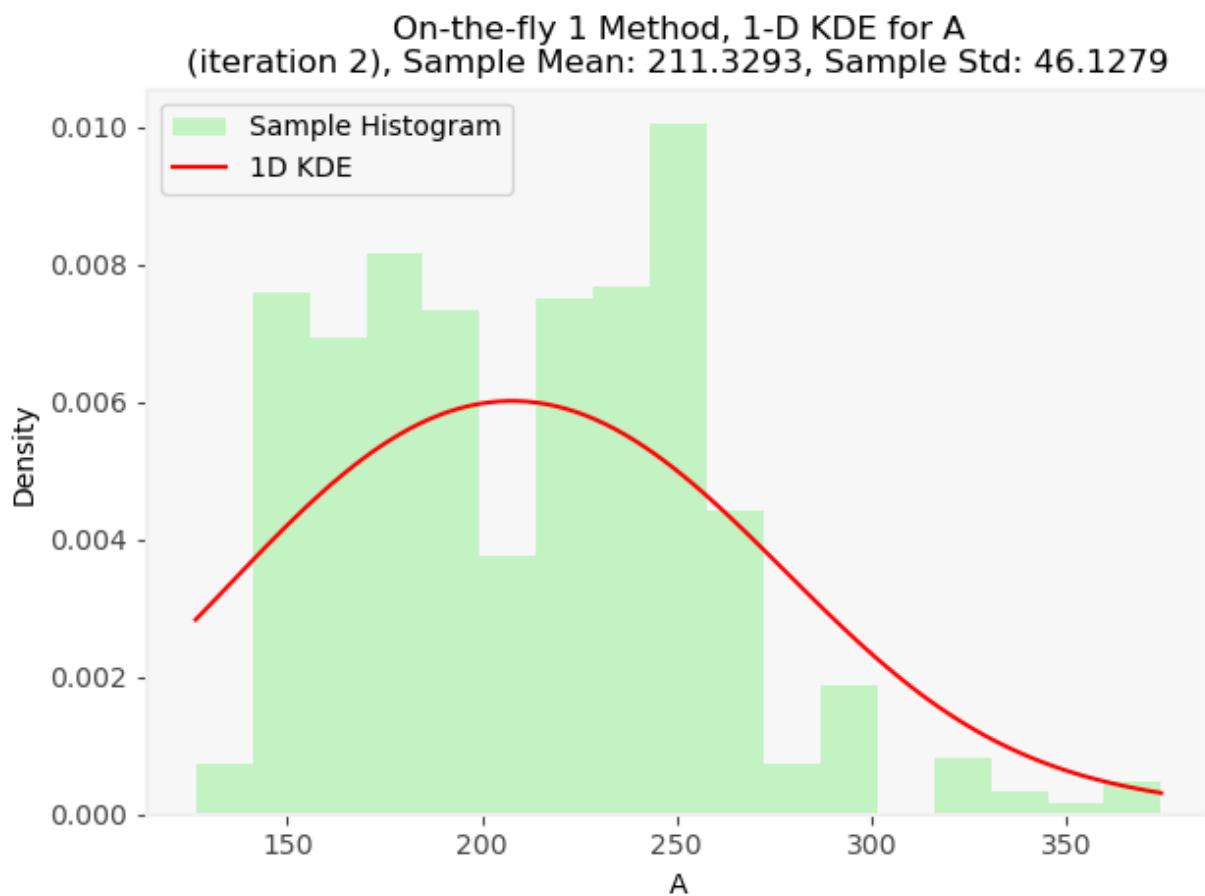
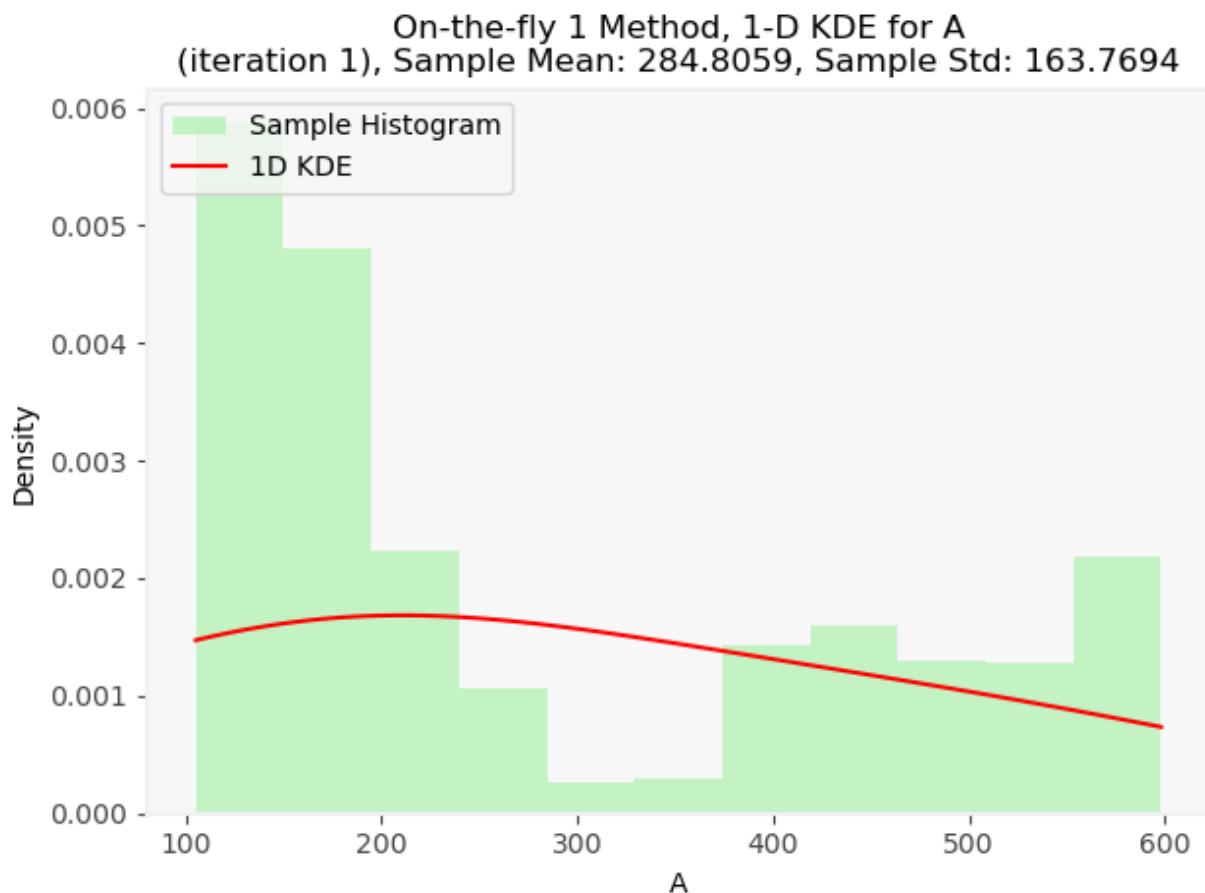


Entropy 2 Method, 1-D KDE for A
(iteration 19), Sample Mean: 151.9052, Sample Std: 2.0981

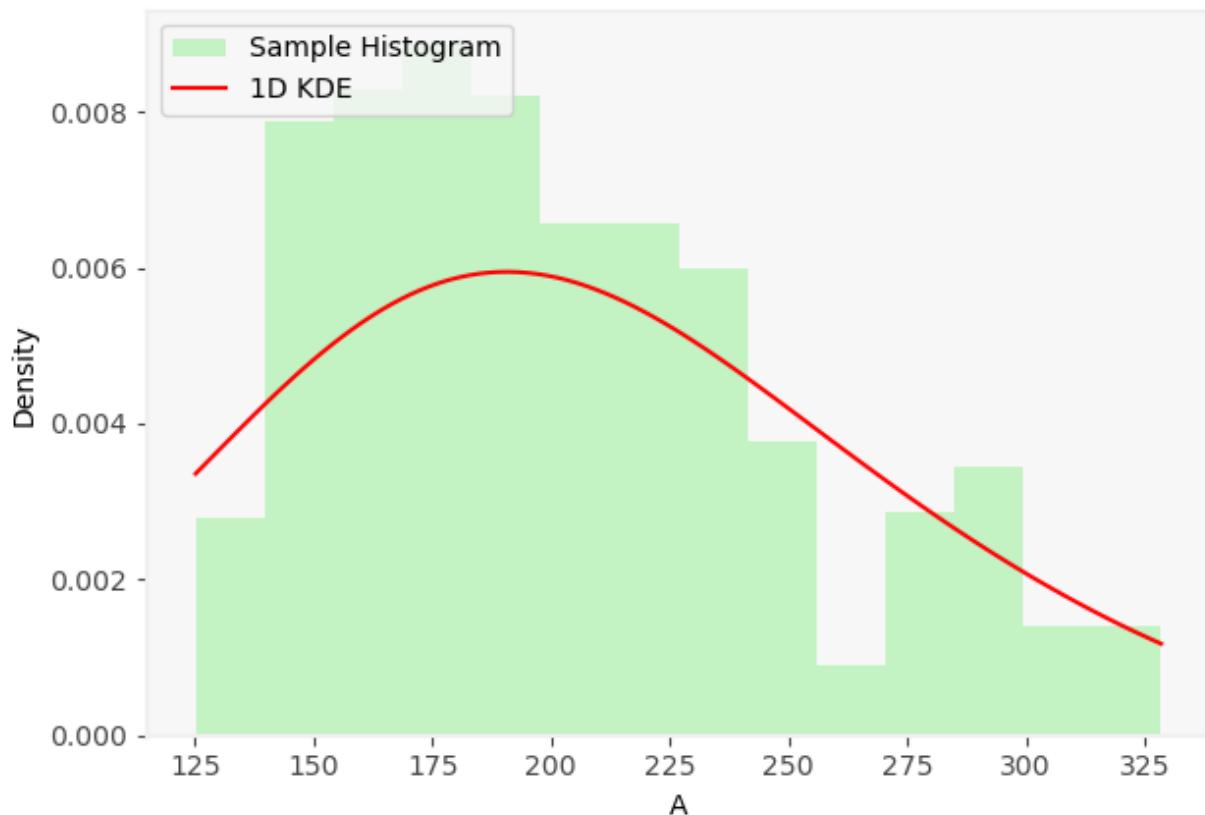


```
In [54]: #Printing the evolution of parameter A for On-the-fly 1
for i in range(len(exp_on_the_fly1.totaltimes())):
    MyPlots.plot_hist_1d_kde(list_par_separated_o1[0][i], kdes_on_the_fly1[i,0],"On-the-fly 1 Method, 1-D KDE for A (iteration 0), Sample Mean: 315.0750, Sample Std: 166.0957")
```

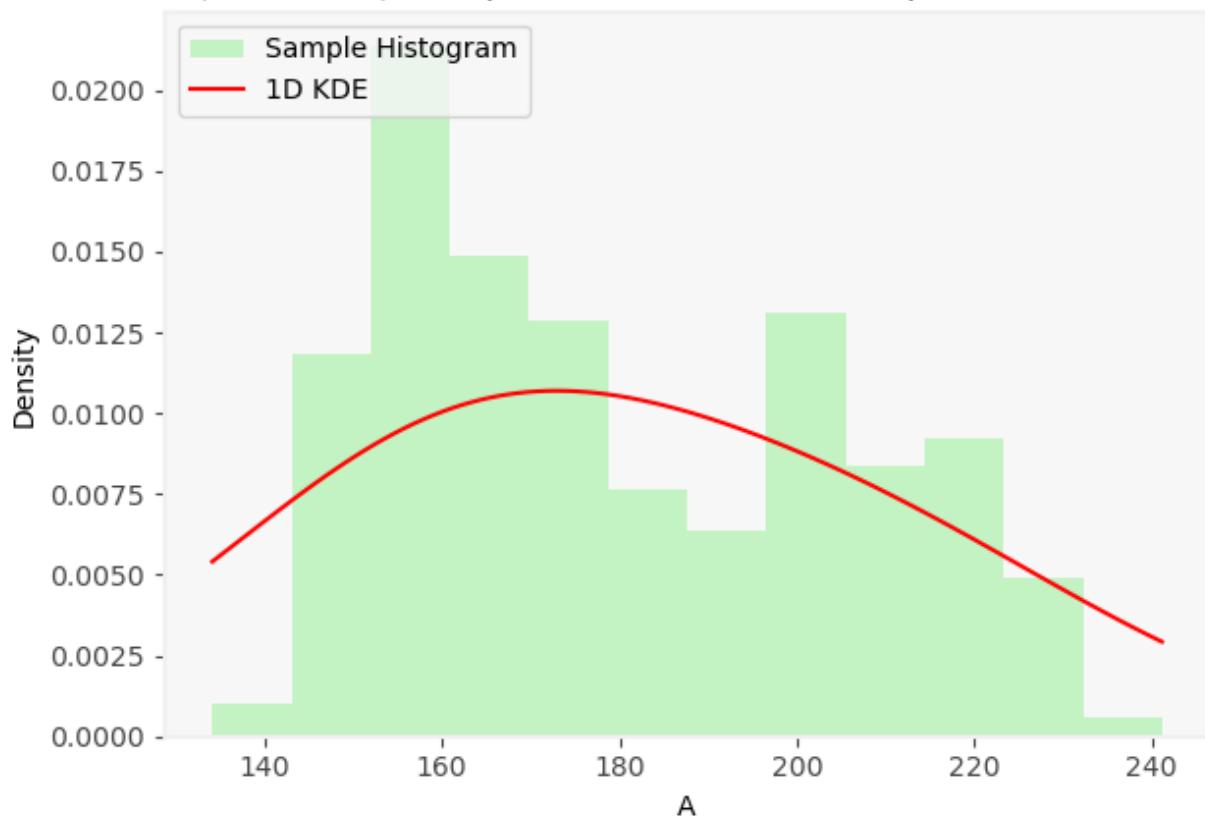




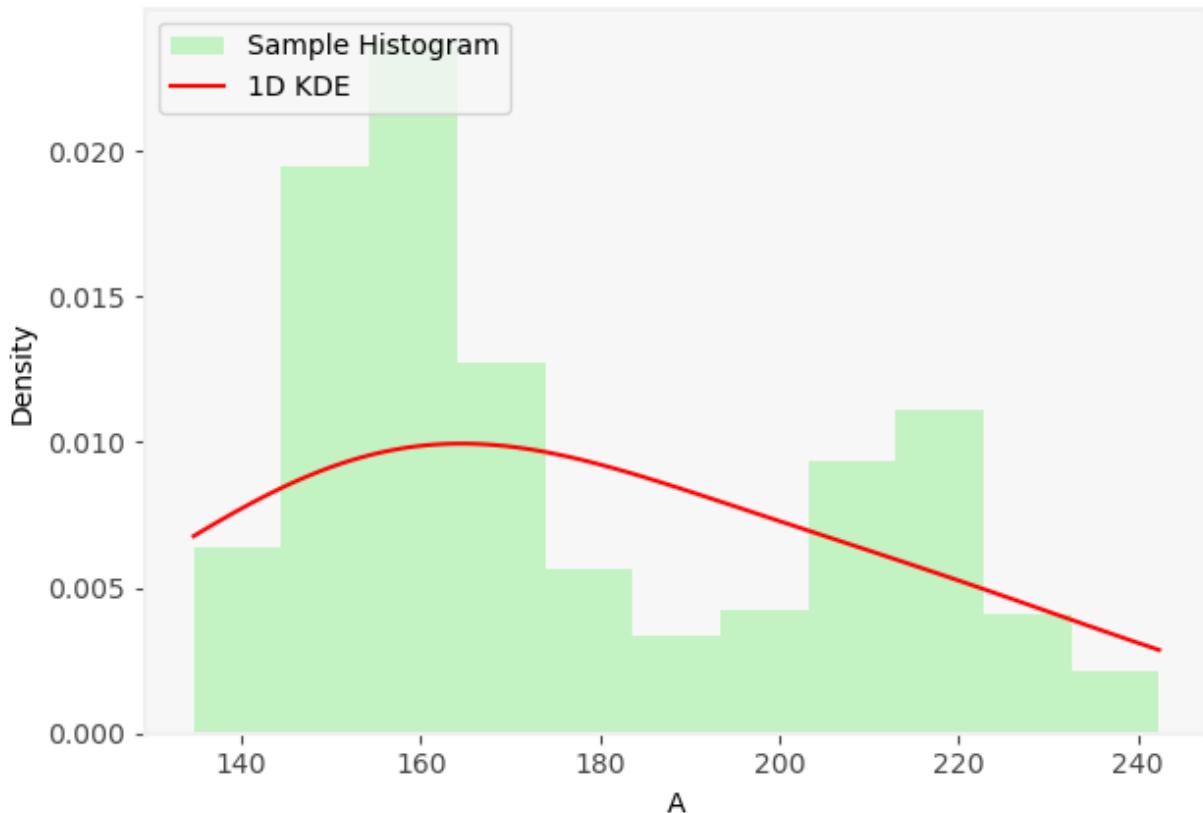
On-the-fly 1 Method, 1-D KDE for A
(iteration 3), Sample Mean: 203.1425, Sample Std: 47.4452



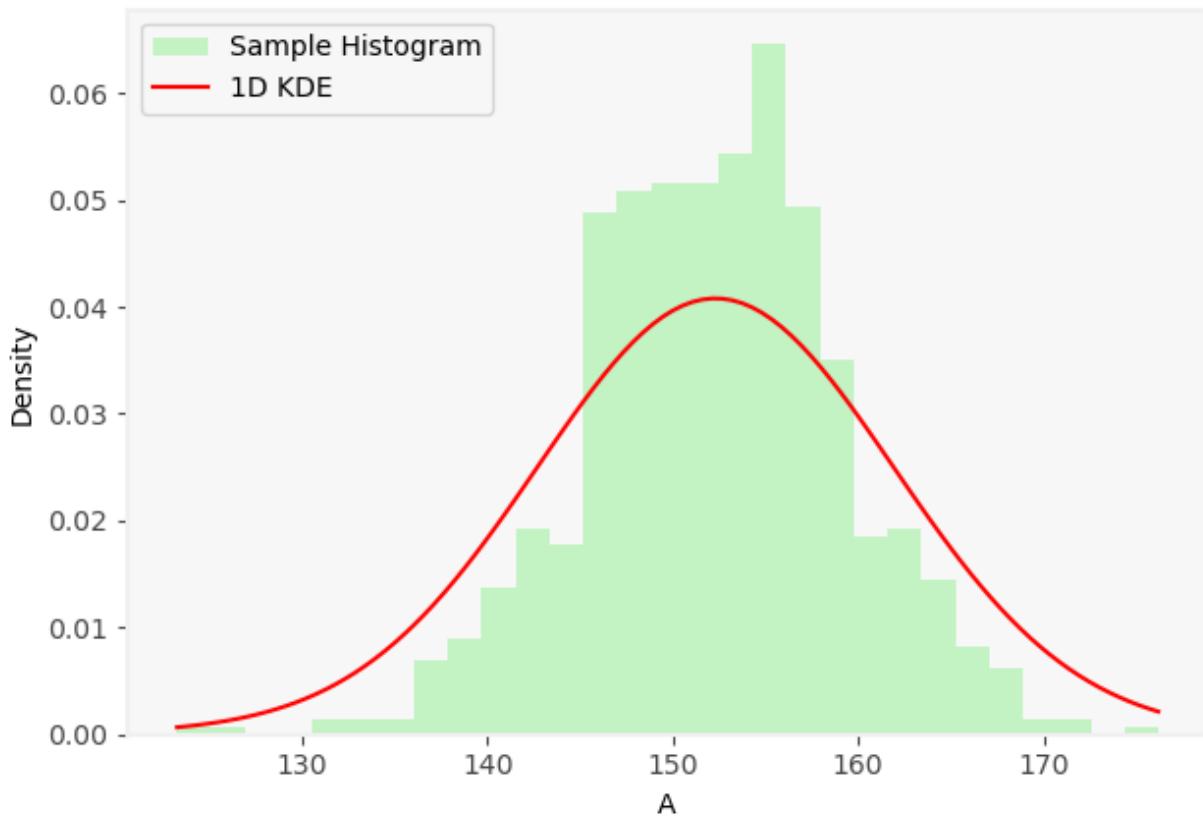
On-the-fly 1 Method, 1-D KDE for A
(iteration 4), Sample Mean: 180.2722, Sample Std: 25.1377



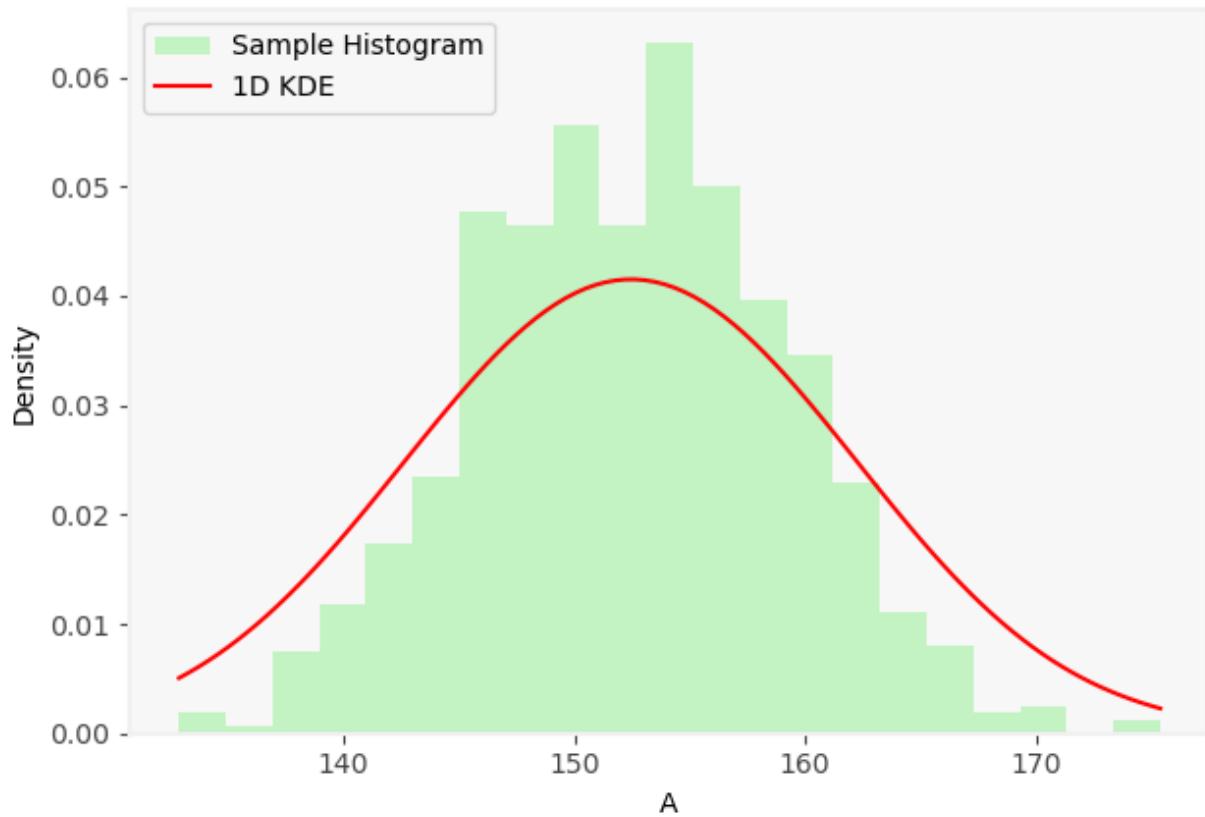
On-the-fly 1 Method, 1-D KDE for A
(iteration 5), Sample Mean: 176.0444, Sample Std: 27.9769



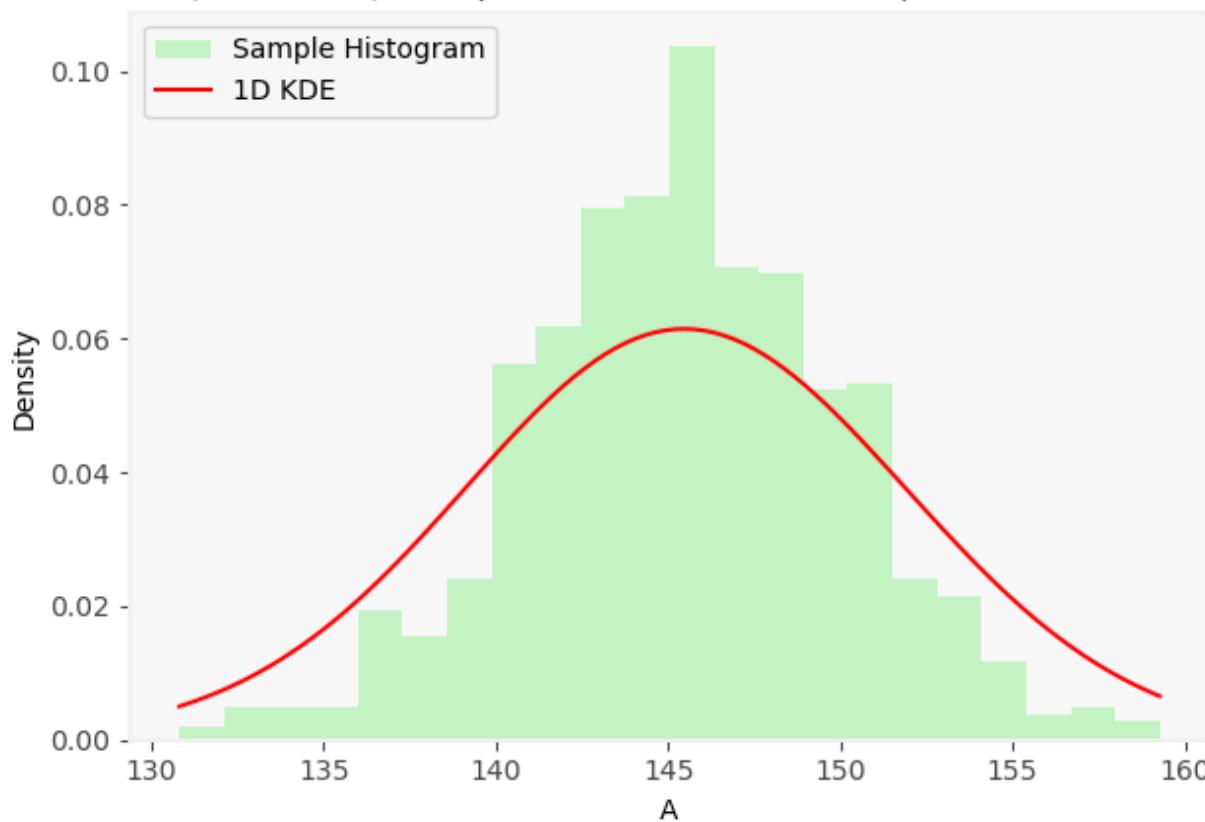
On-the-fly 1 Method, 1-D KDE for A
(iteration 6), Sample Mean: 152.2302, Sample Std: 7.0010



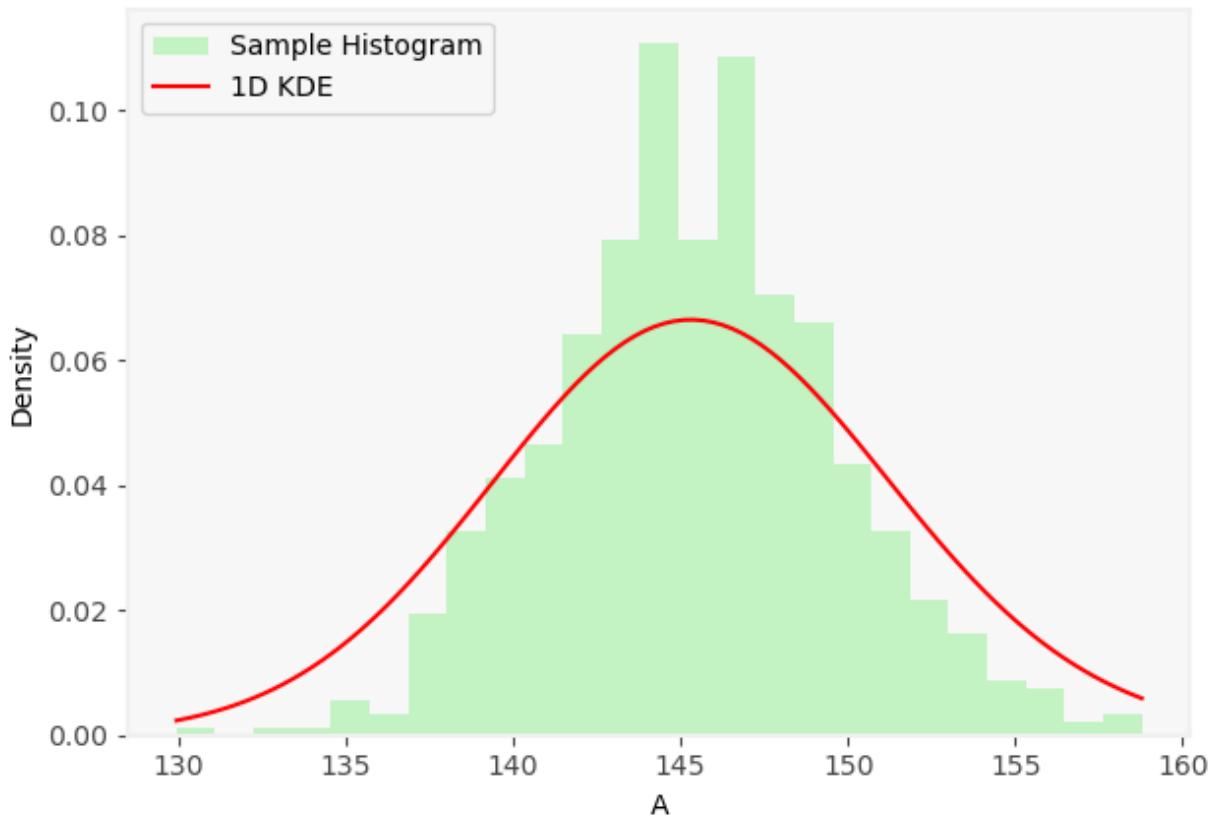
On-the-fly 1 Method, 1-D KDE for A
(iteration 7), Sample Mean: 152.4600, Sample Std: 6.7362



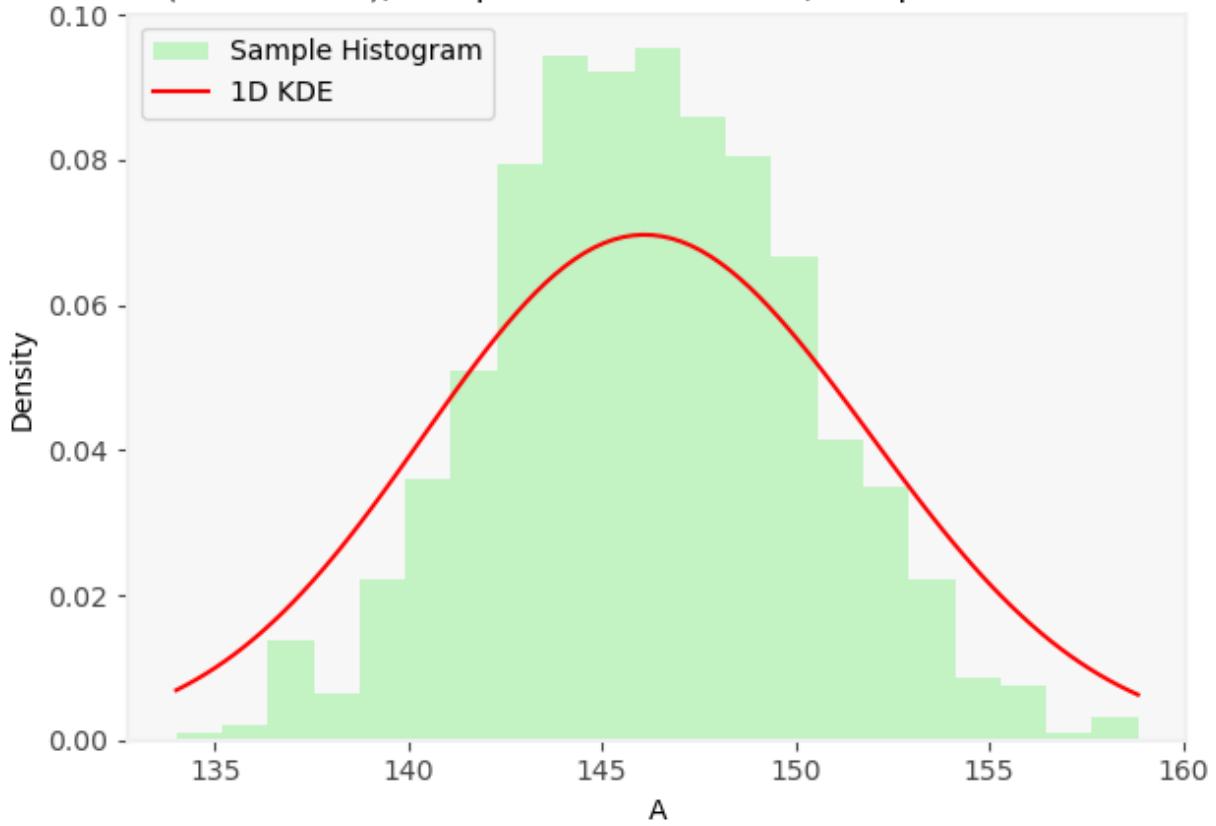
On-the-fly 1 Method, 1-D KDE for A
(iteration 8), Sample Mean: 145.4696, Sample Std: 4.6193



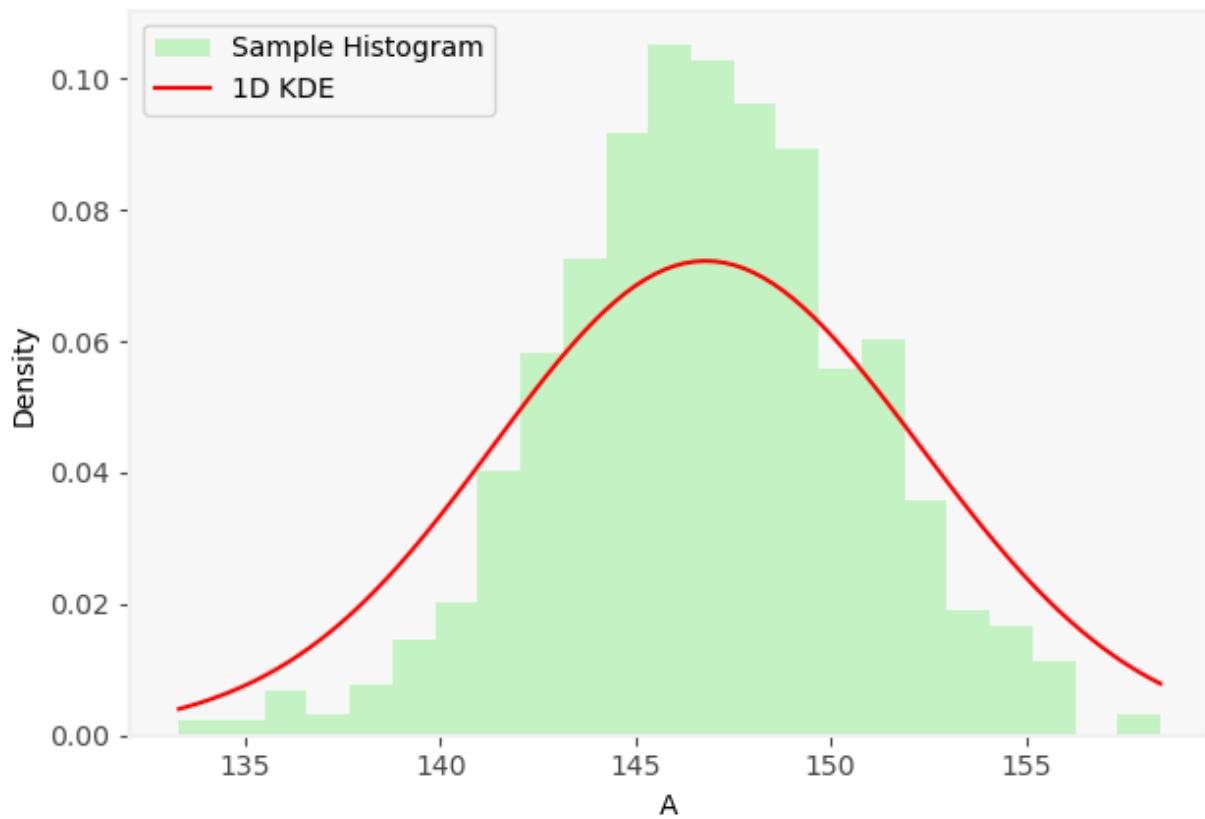
On-the-fly 1 Method, 1-D KDE for A
(iteration 9), Sample Mean: 145.4015, Sample Std: 4.2711



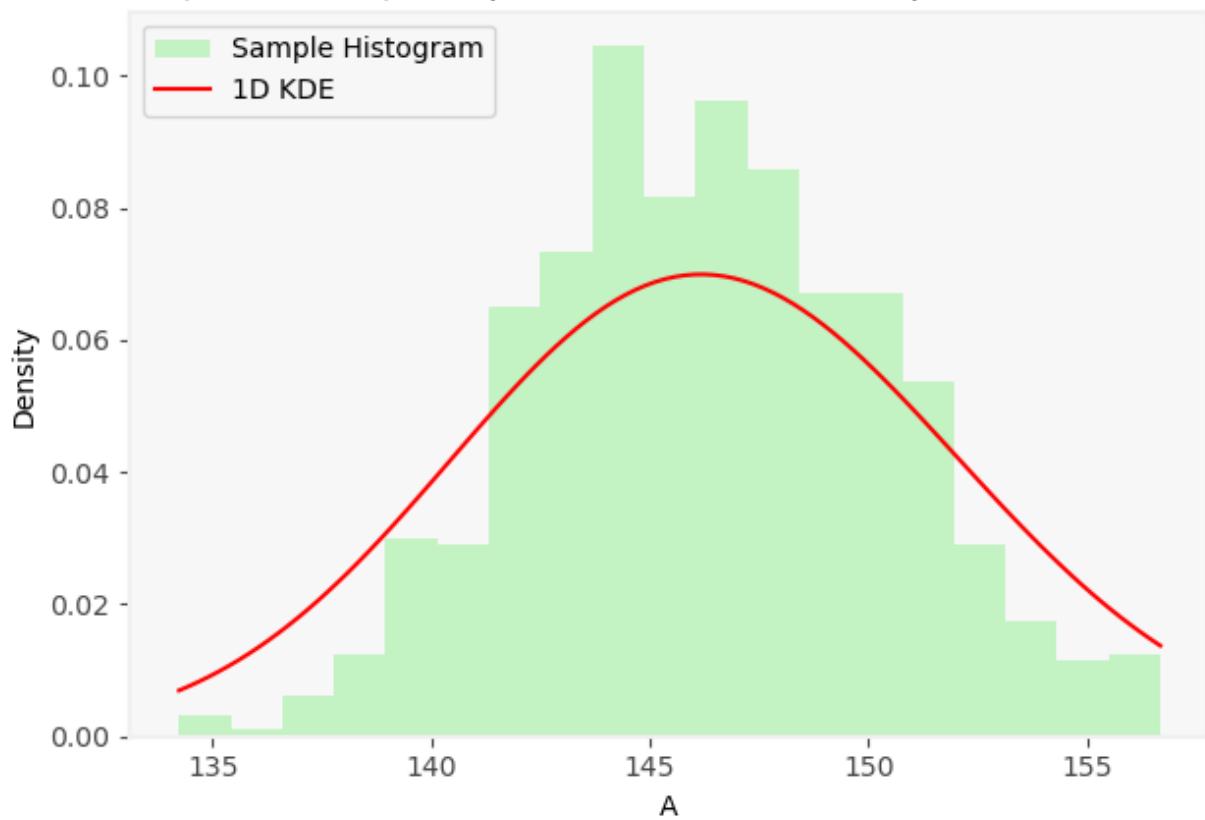
On-the-fly 1 Method, 1-D KDE for A
(iteration 10), Sample Mean: 146.2337, Sample Std: 4.0348



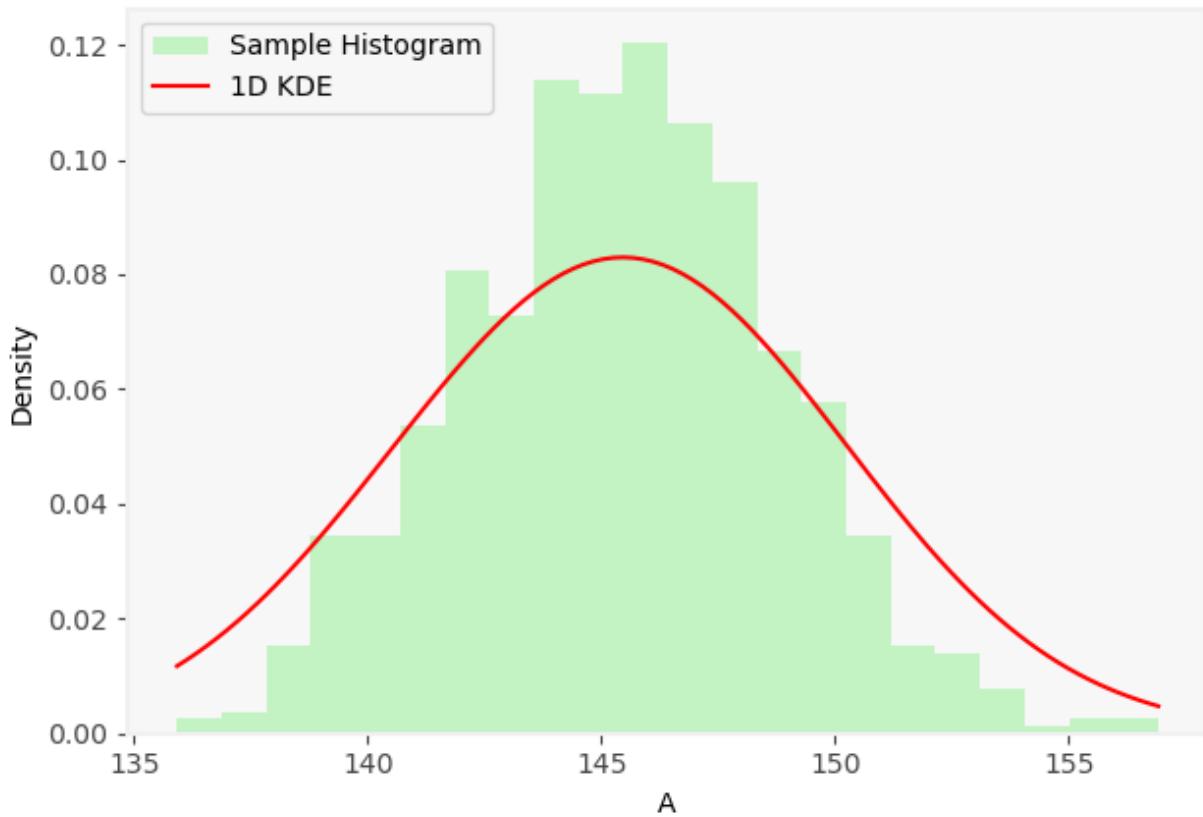
On-the-fly 1 Method, 1-D KDE for A
(iteration 11), Sample Mean: 146.7735, Sample Std: 3.9331



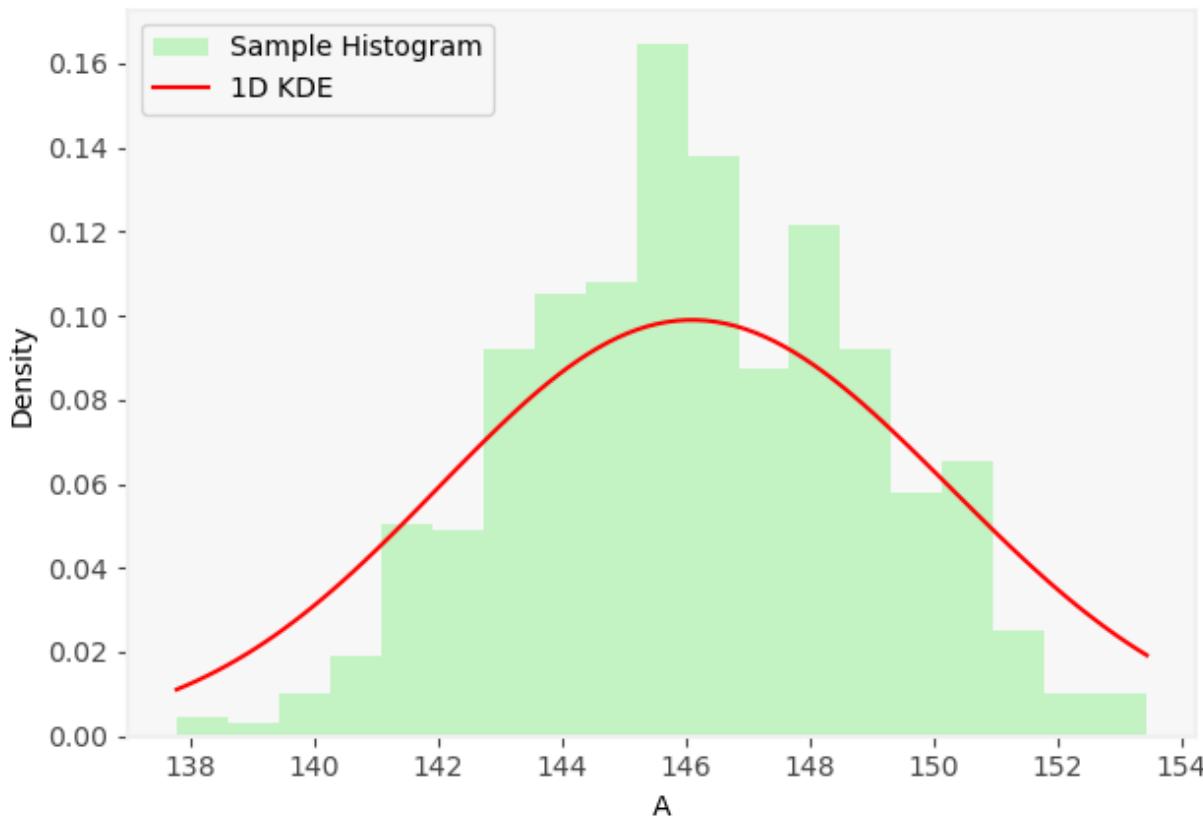
On-the-fly 1 Method, 1-D KDE for A
(iteration 12), Sample Mean: 146.3419, Sample Std: 3.9969

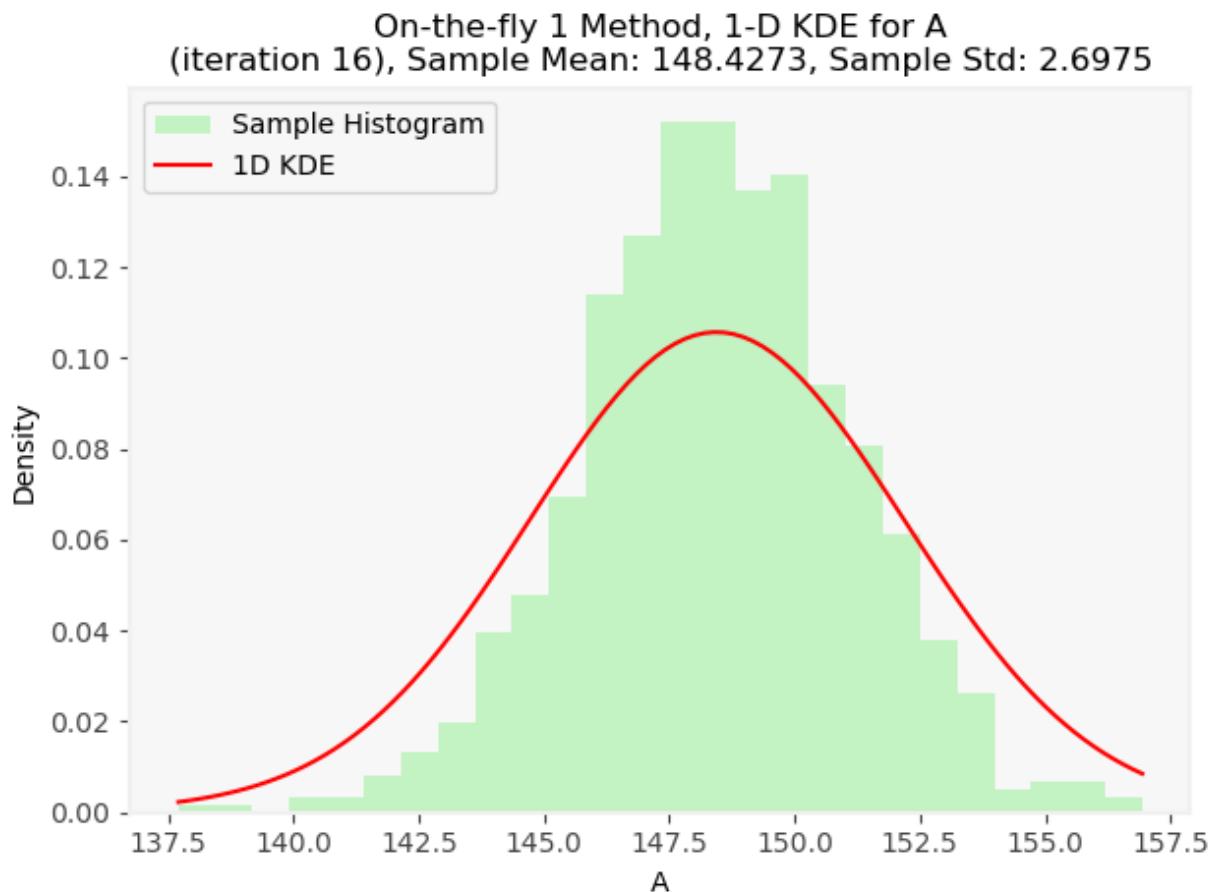
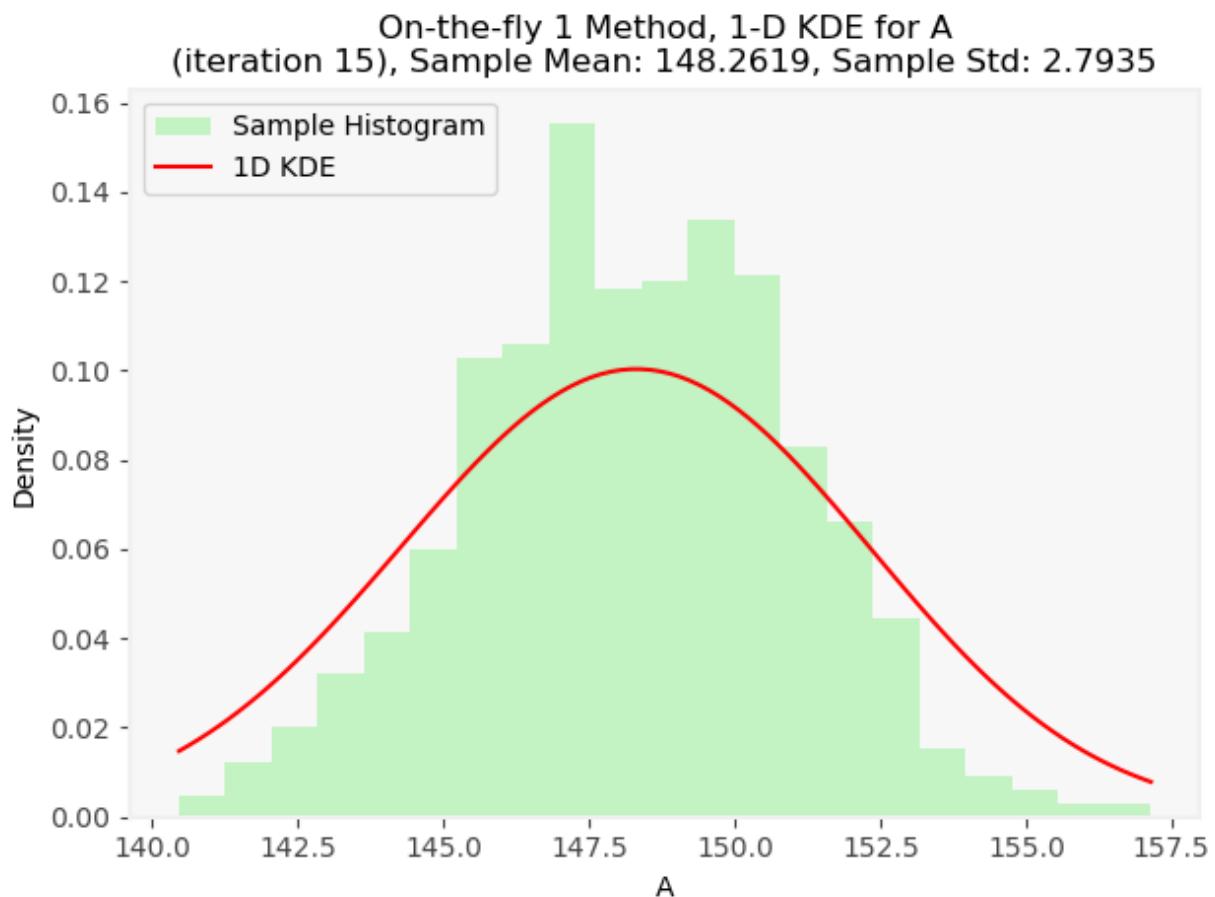


On-the-fly 1 Method, 1-D KDE for A
(iteration 13), Sample Mean: 145.4323, Sample Std: 3.3839

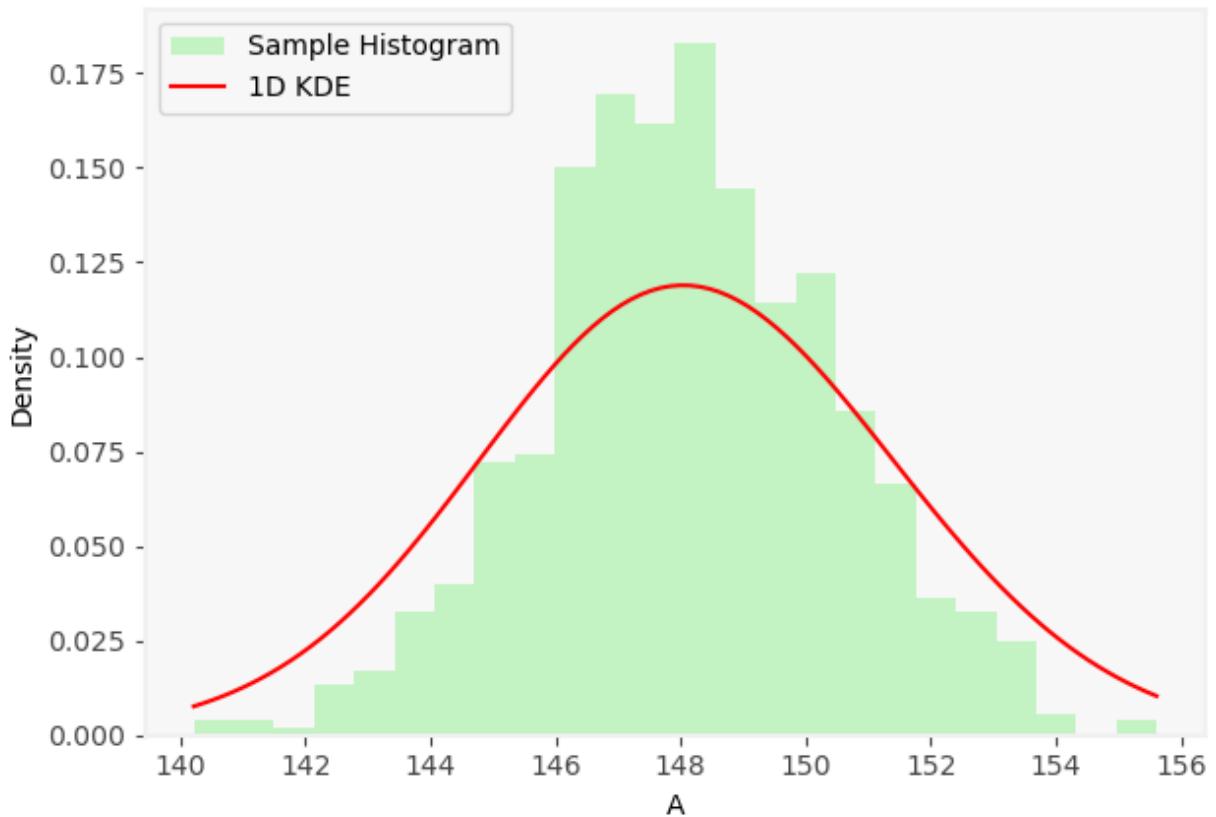


On-the-fly 1 Method, 1-D KDE for A
(iteration 14), Sample Mean: 146.1351, Sample Std: 2.8123

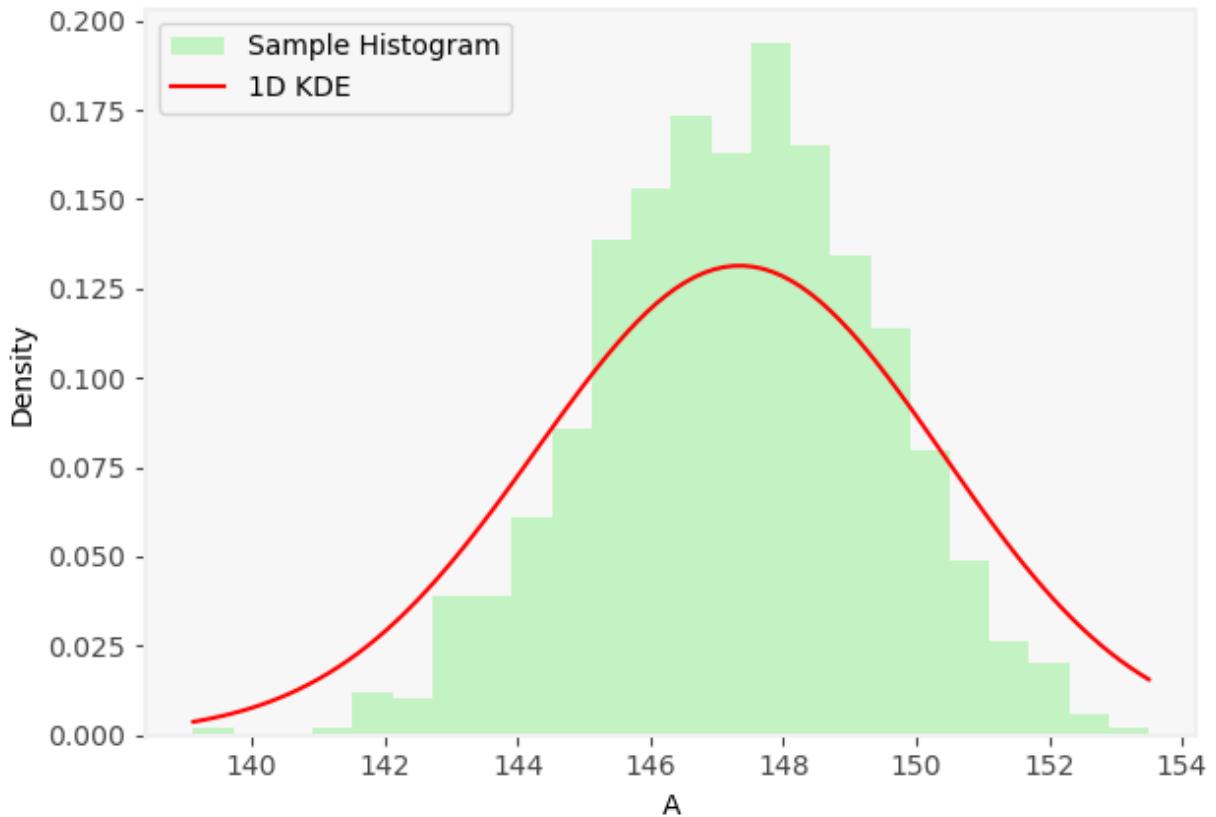




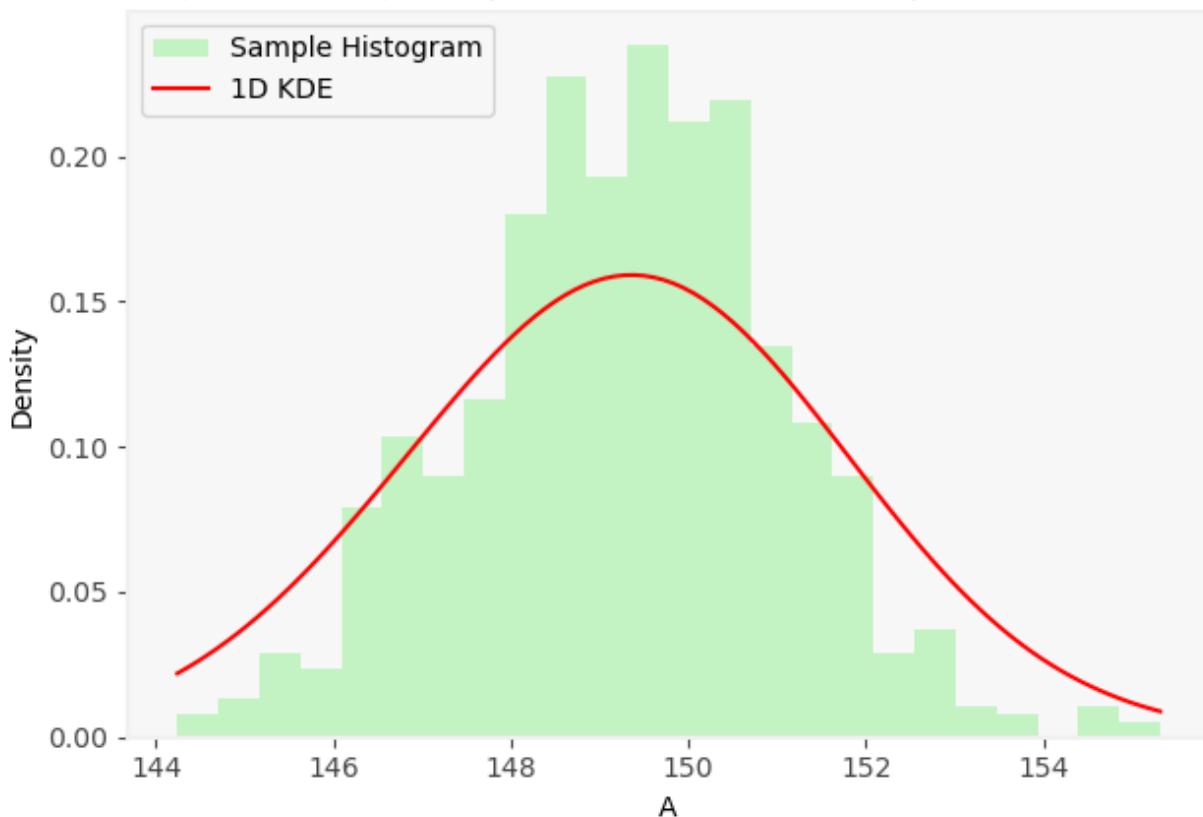
On-the-fly 1 Method, 1-D KDE for A
(iteration 17), Sample Mean: 148.1133, Sample Std: 2.3880



On-the-fly 1 Method, 1-D KDE for A
(iteration 18), Sample Mean: 147.2714, Sample Std: 2.1395

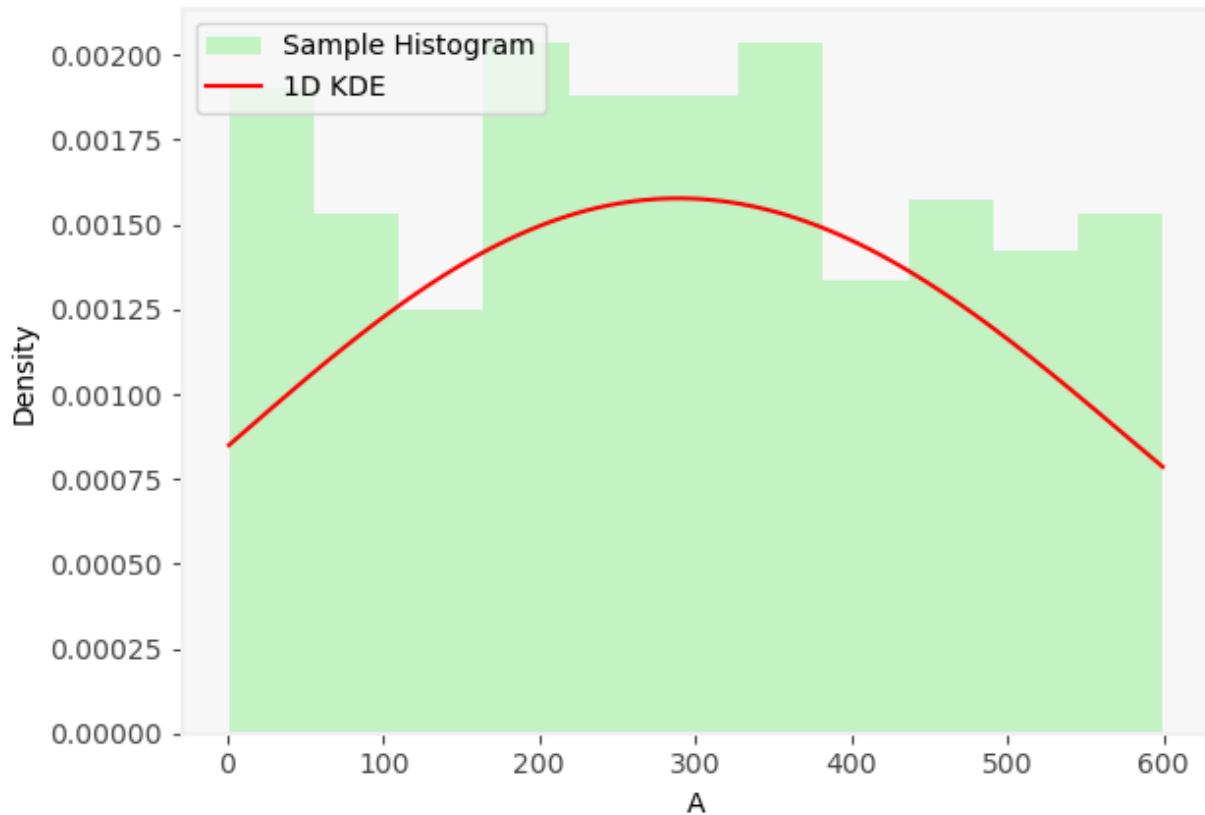


On-the-fly 1 Method, 1-D KDE for A
(iteration 19), Sample Mean: 149.2862, Sample Std: 1.7829

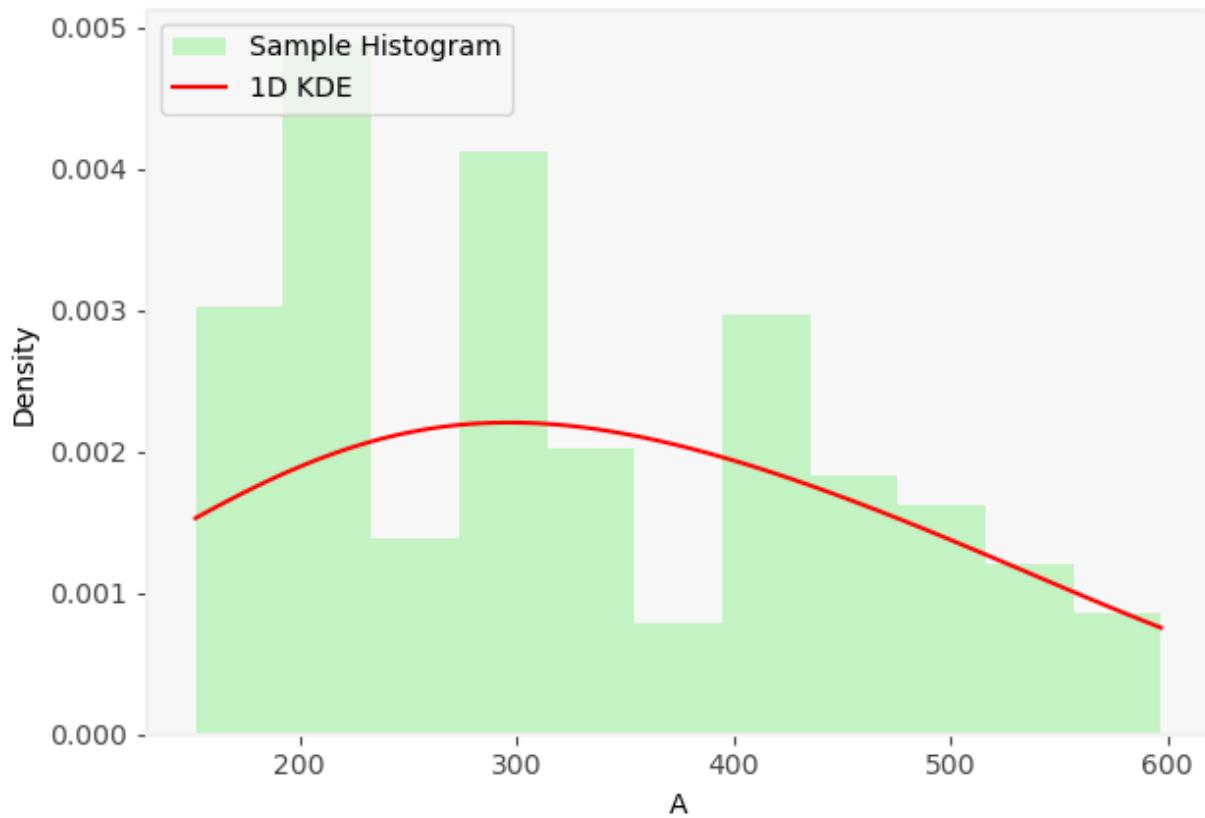


```
In [55]: #Printing the evolution of parameter A for On-the-fly 2
for i in range(len(exp_on_the_fly2.totaltimes())):
    MyPlots.plot_hist_1d_kde(list_par_separated_o2[0][i], kdes_on_the_fly2[i,0],"On-th
```

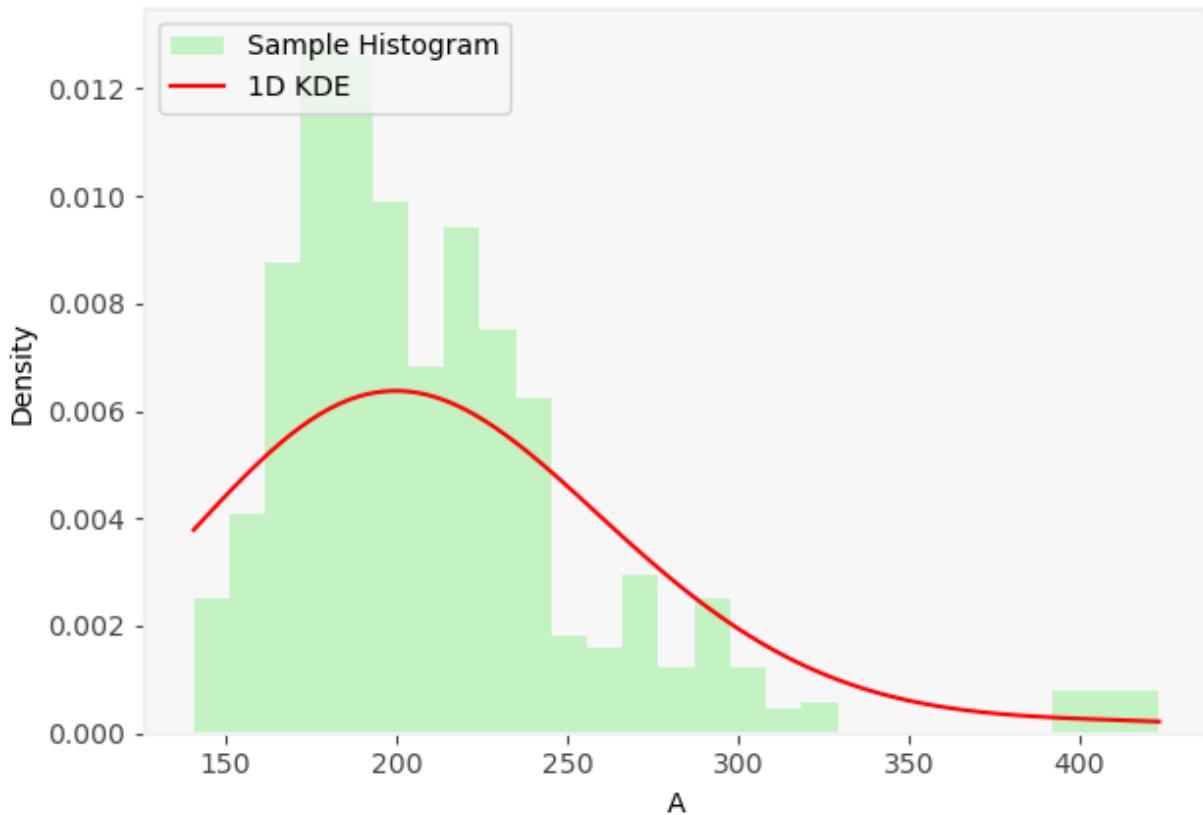
On-the-fly 2 Method, 1-D KDE for A
(iteration 0), Sample Mean: 293.0418, Sample Std: 169.9774



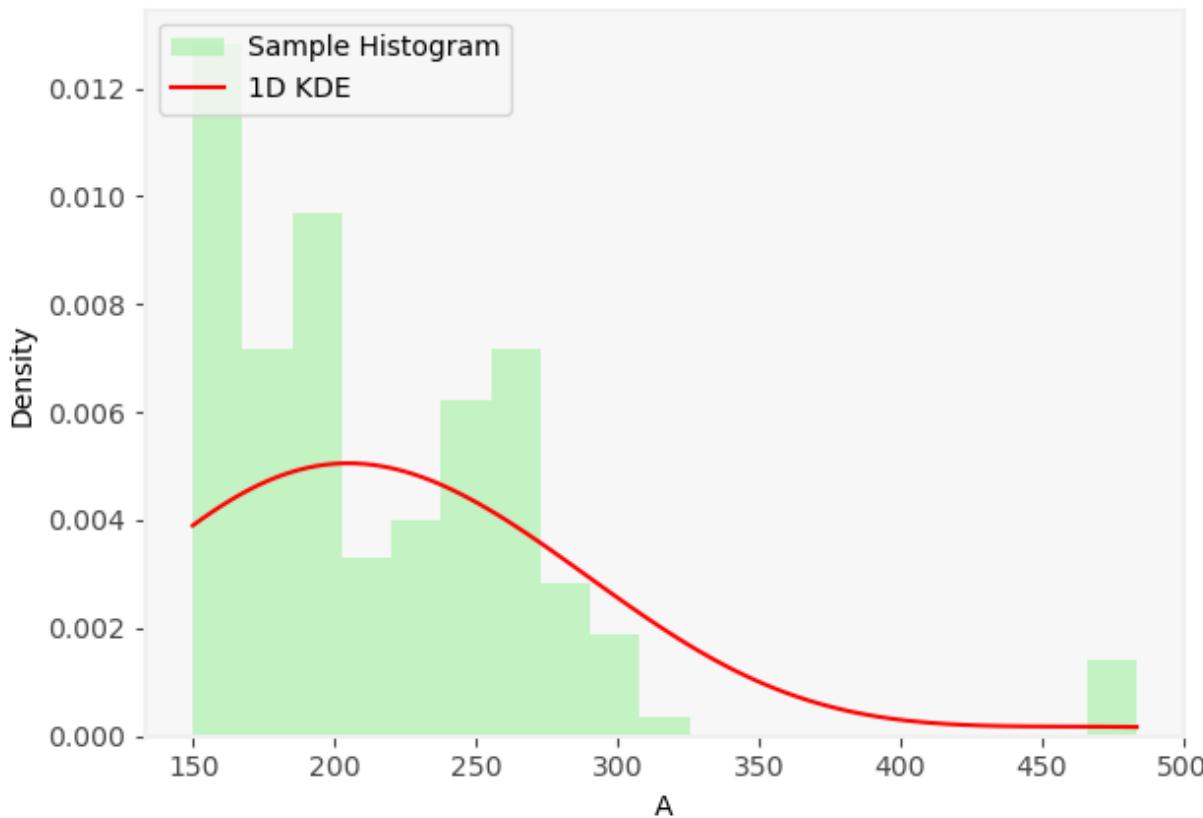
On-the-fly 2 Method, 1-D KDE for A
(iteration 1), Sample Mean: 328.4801, Sample Std: 122.5961



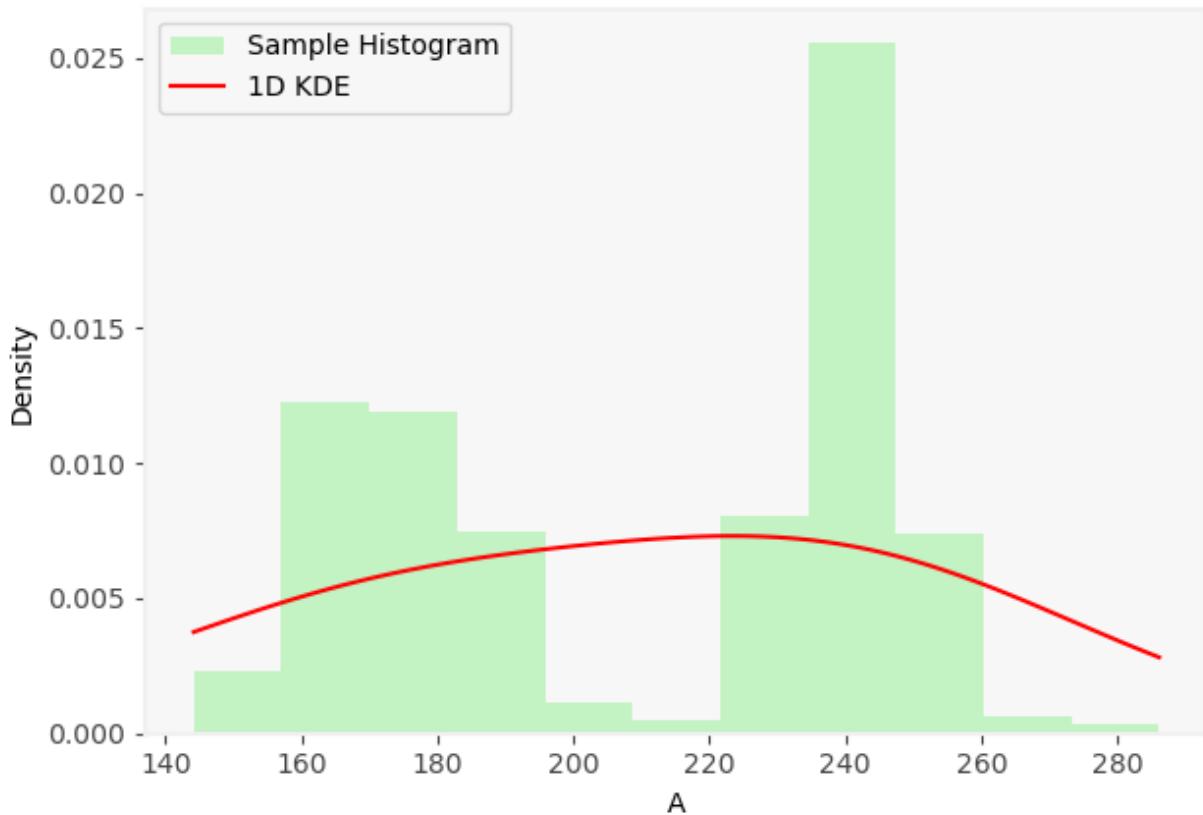
On-the-fly 2 Method, 1-D KDE for A
(iteration 2), Sample Mean: 211.7689, Sample Std: 48.9599



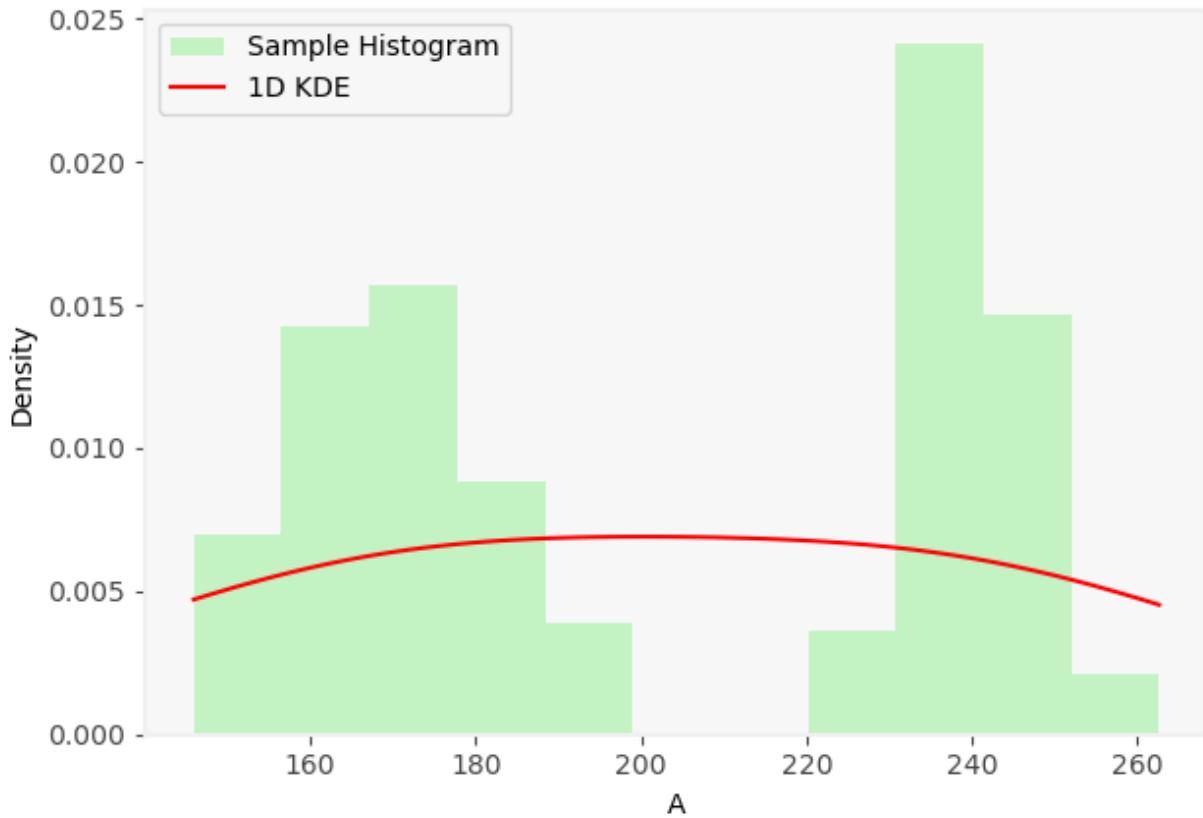
On-the-fly 2 Method, 1-D KDE for A
(iteration 3), Sample Mean: 217.3159, Sample Std: 60.5285



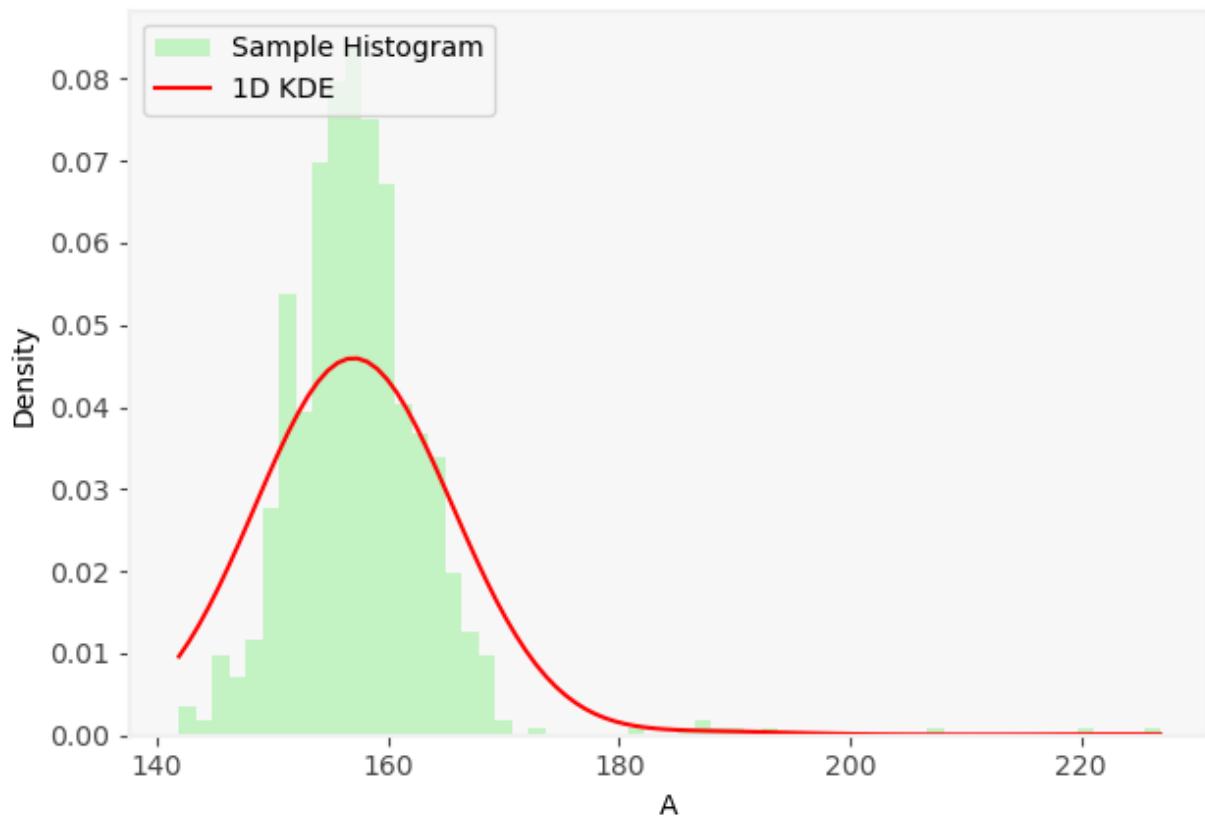
On-the-fly 2 Method, 1-D KDE for A
(iteration 4), Sample Mean: 210.3493, Sample Std: 35.2895



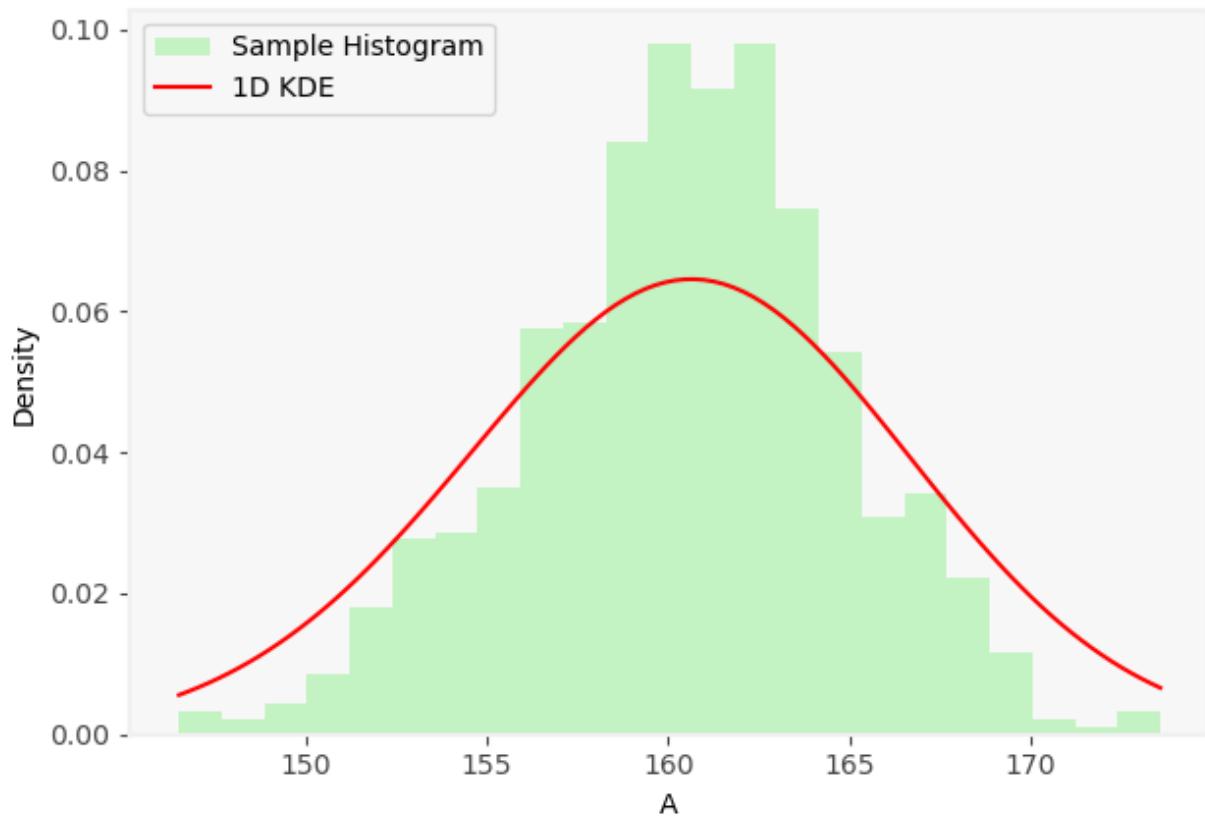
On-the-fly 2 Method, 1-D KDE for A
(iteration 5), Sample Mean: 203.2103, Sample Std: 36.1650



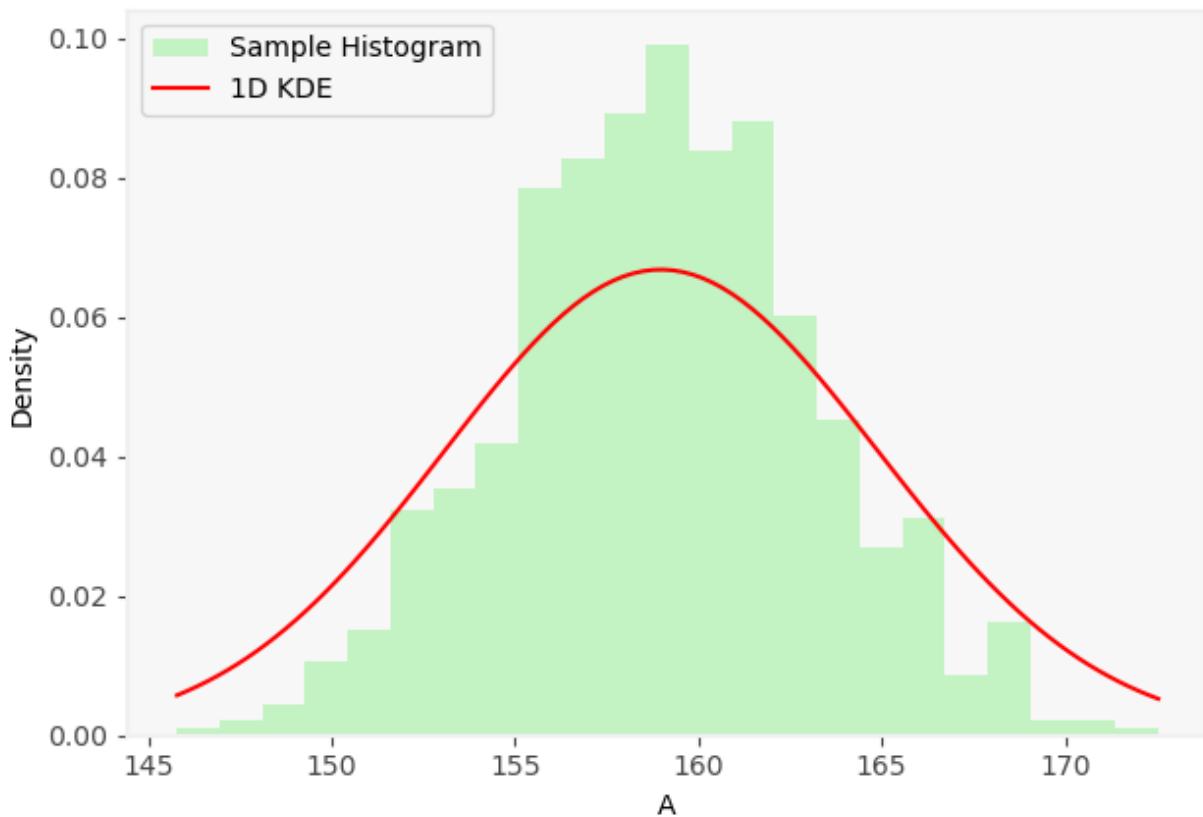
On-the-fly 2 Method, 1-D KDE for A
(iteration 6), Sample Mean: 157.6006, Sample Std: 6.9288



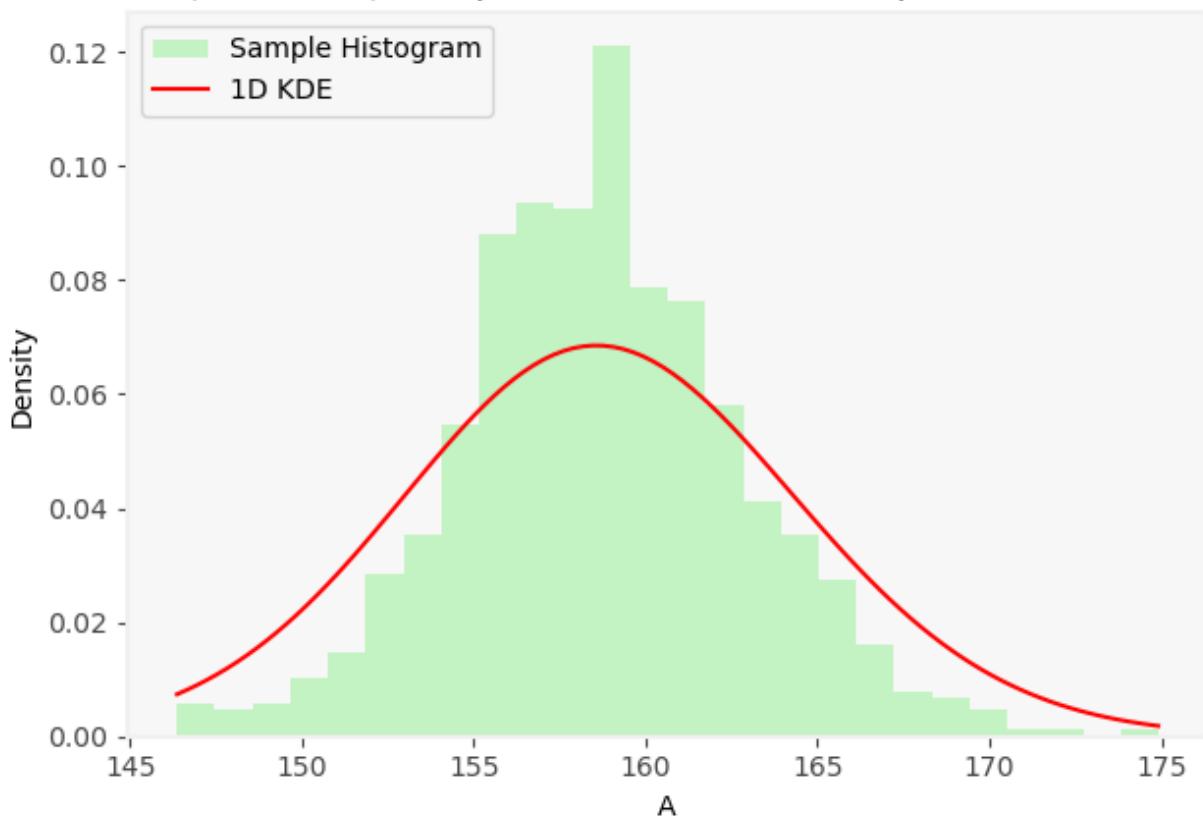
On-the-fly 2 Method, 1-D KDE for A
(iteration 7), Sample Mean: 160.4194, Sample Std: 4.3979



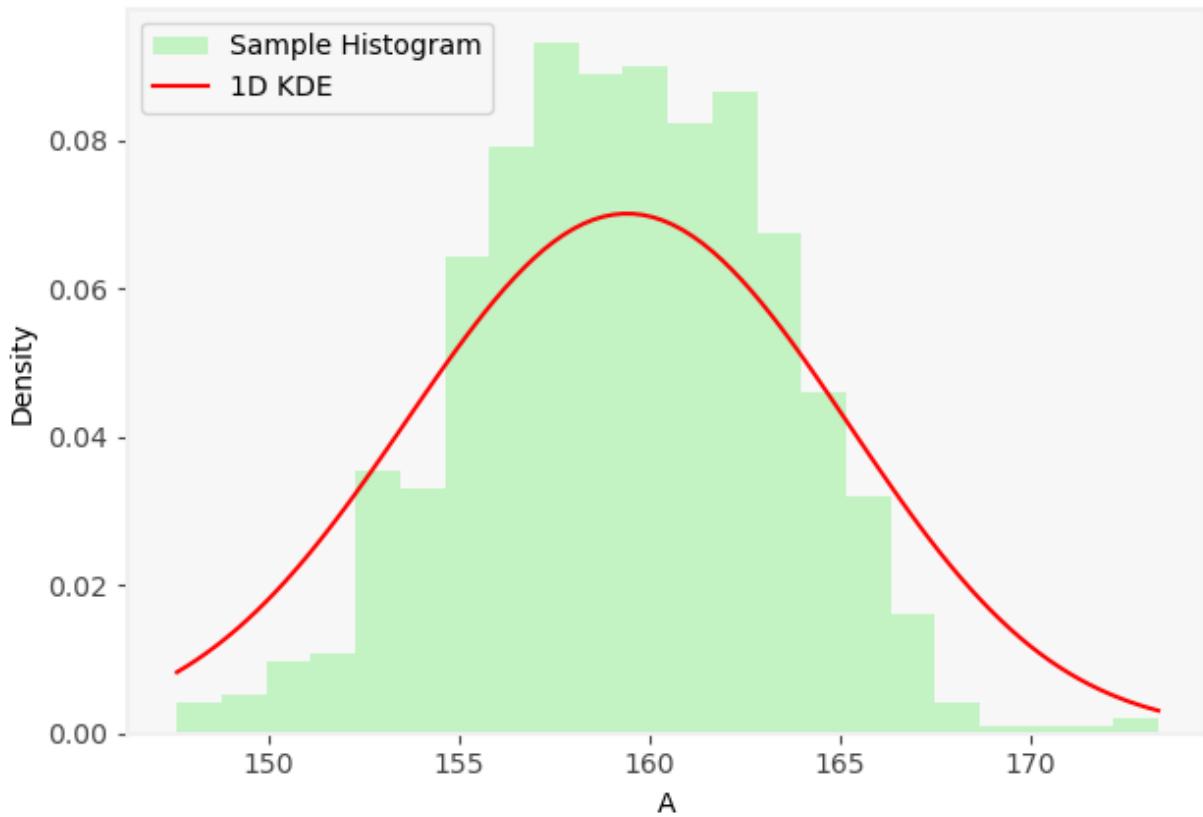
On-the-fly 2 Method, 1-D KDE for A
(iteration 8), Sample Mean: 159.0076, Sample Std: 4.2247



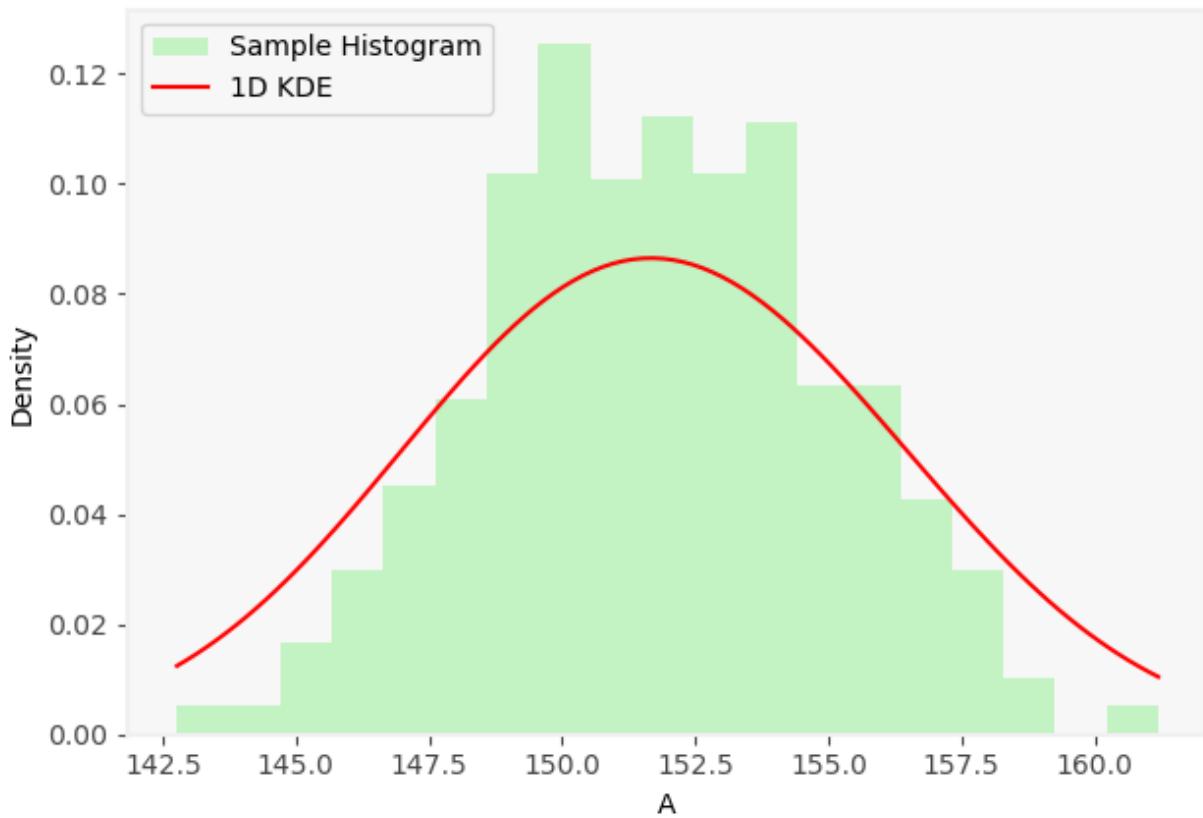
On-the-fly 2 Method, 1-D KDE for A
(iteration 9), Sample Mean: 158.7334, Sample Std: 4.1861



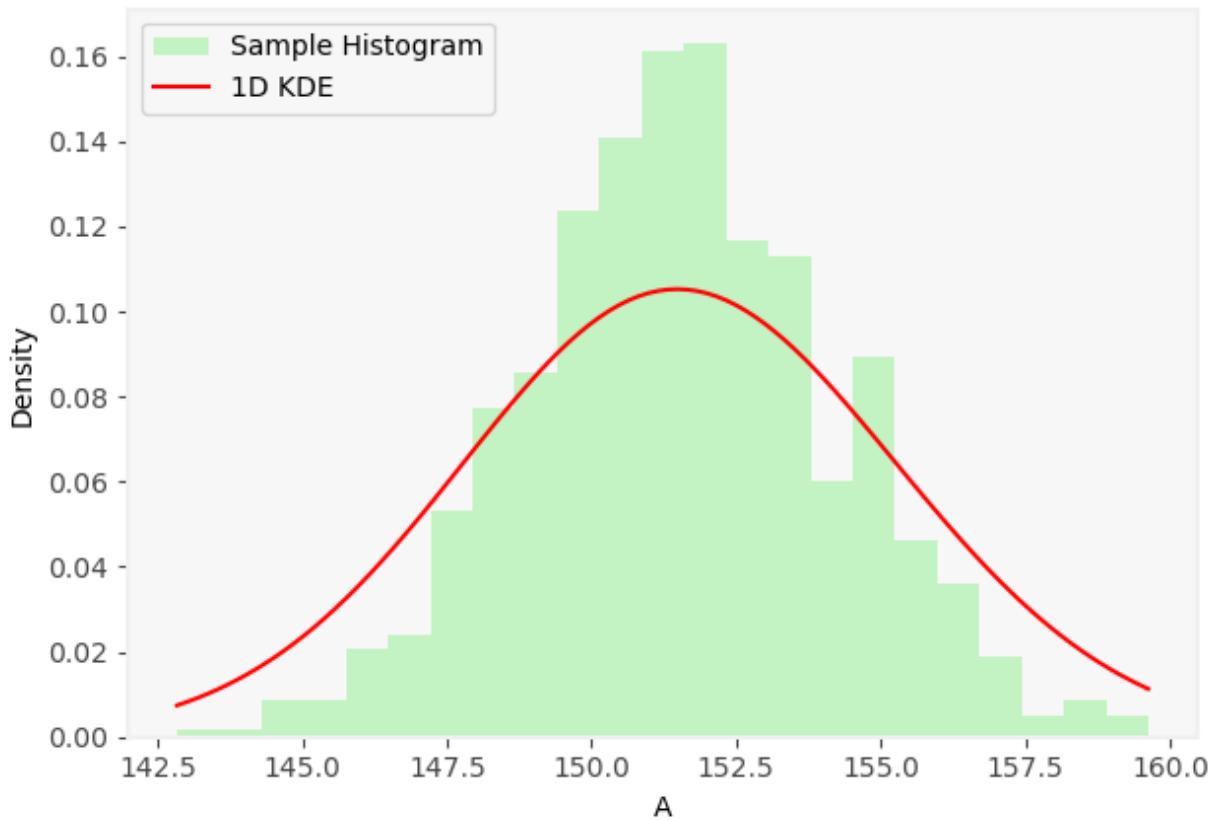
On-the-fly 2 Method, 1-D KDE for A
(iteration 10), Sample Mean: 159.3606, Sample Std: 3.9994



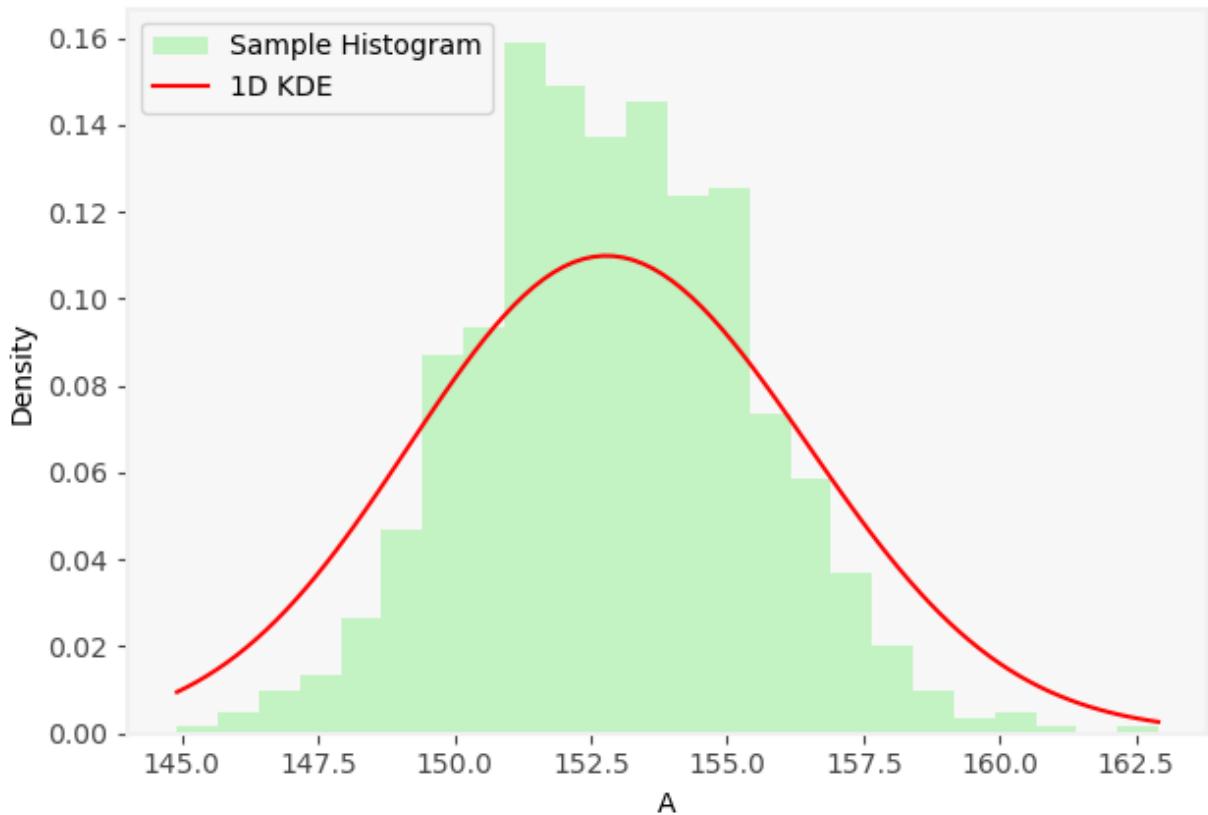
On-the-fly 2 Method, 1-D KDE for A
(iteration 11), Sample Mean: 151.7609, Sample Std: 3.2133



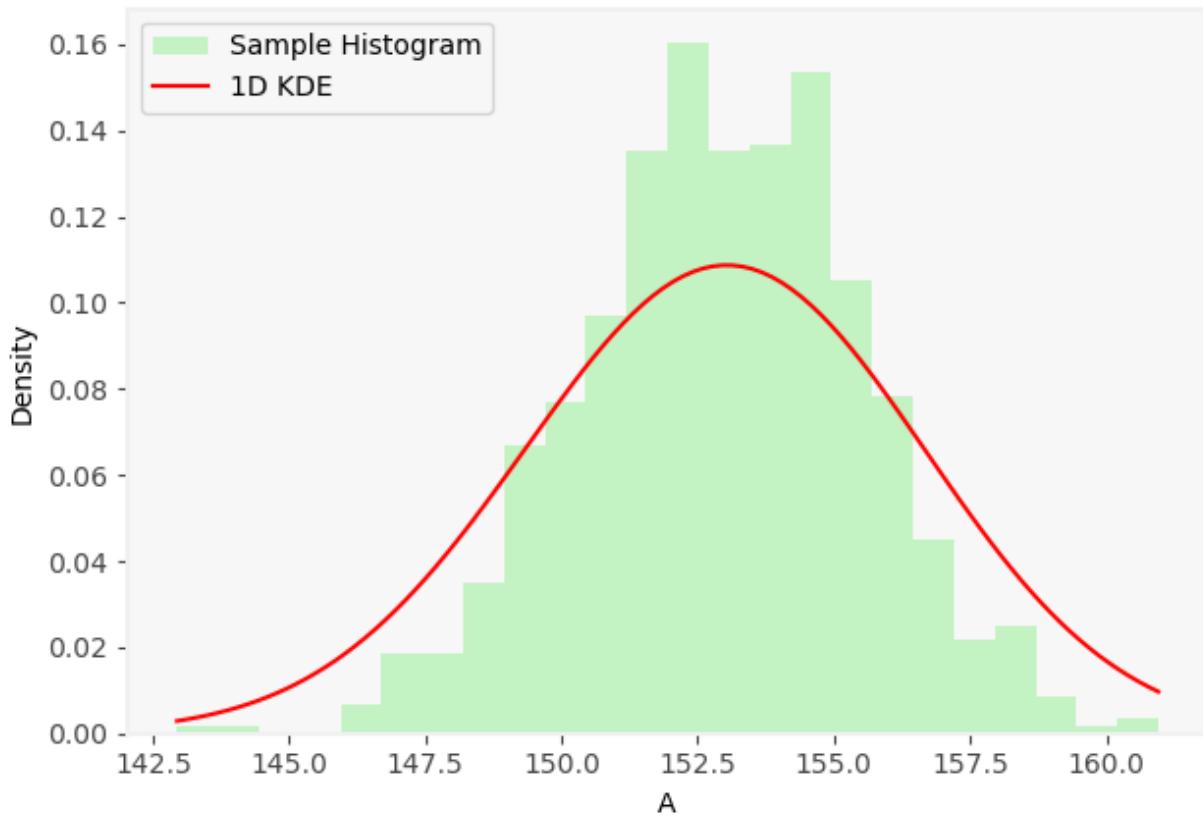
On-the-fly 2 Method, 1-D KDE for A
(iteration 12), Sample Mean: 151.5586, Sample Std: 2.6880



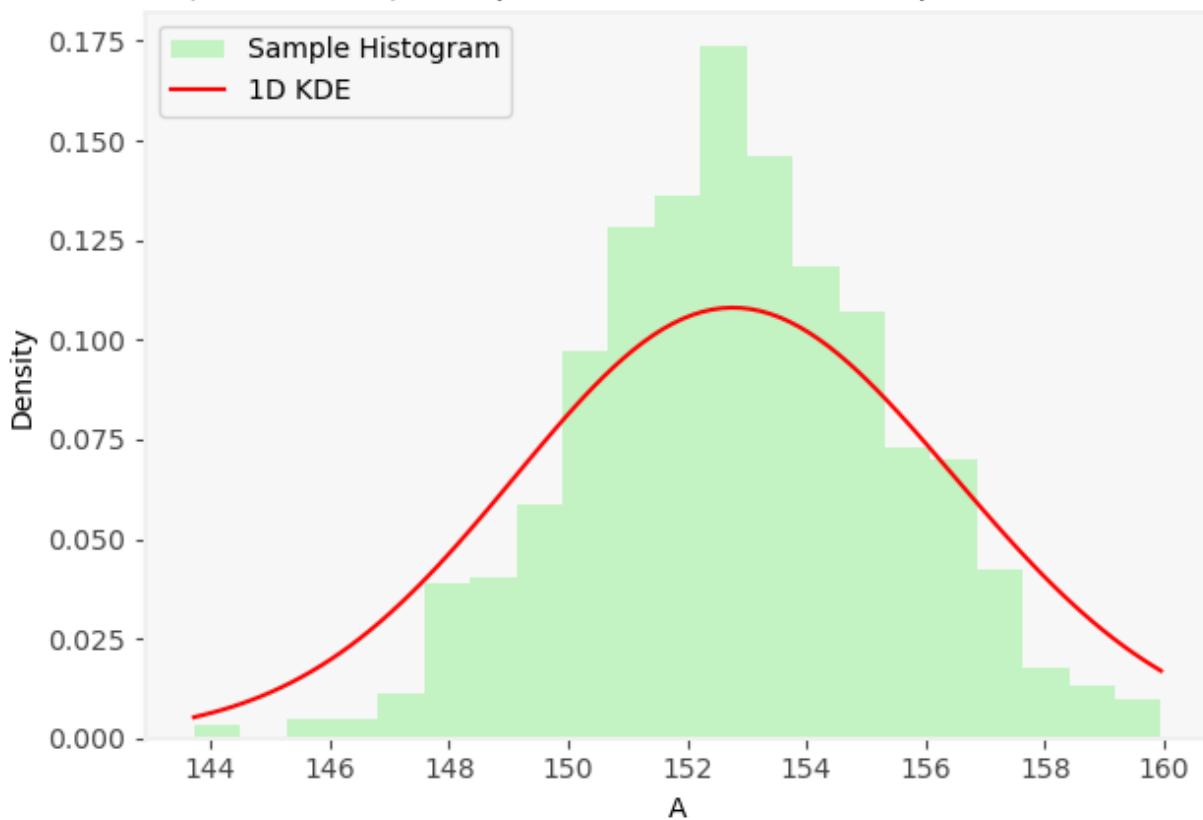
On-the-fly 2 Method, 1-D KDE for A
(iteration 13), Sample Mean: 152.8789, Sample Std: 2.5664



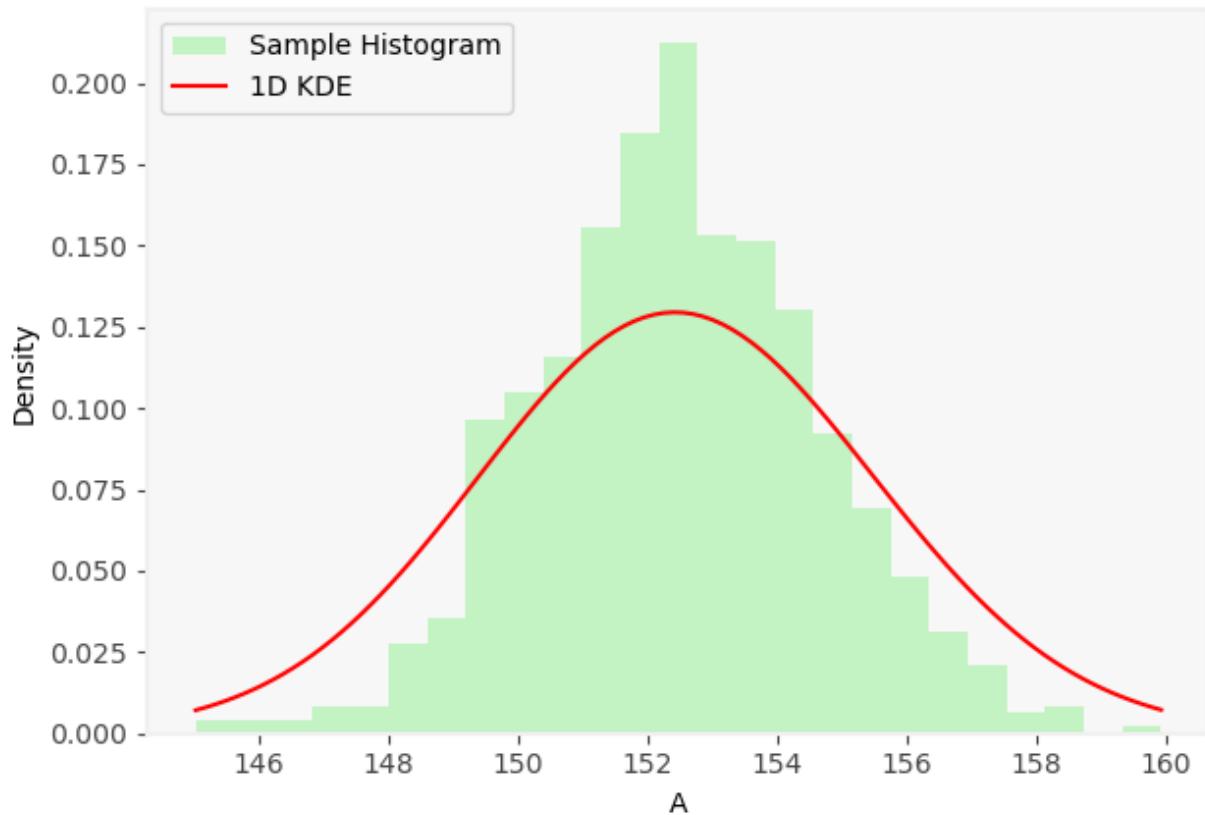
On-the-fly 2 Method, 1-D KDE for A
(iteration 14), Sample Mean: 152.9386, Sample Std: 2.5979



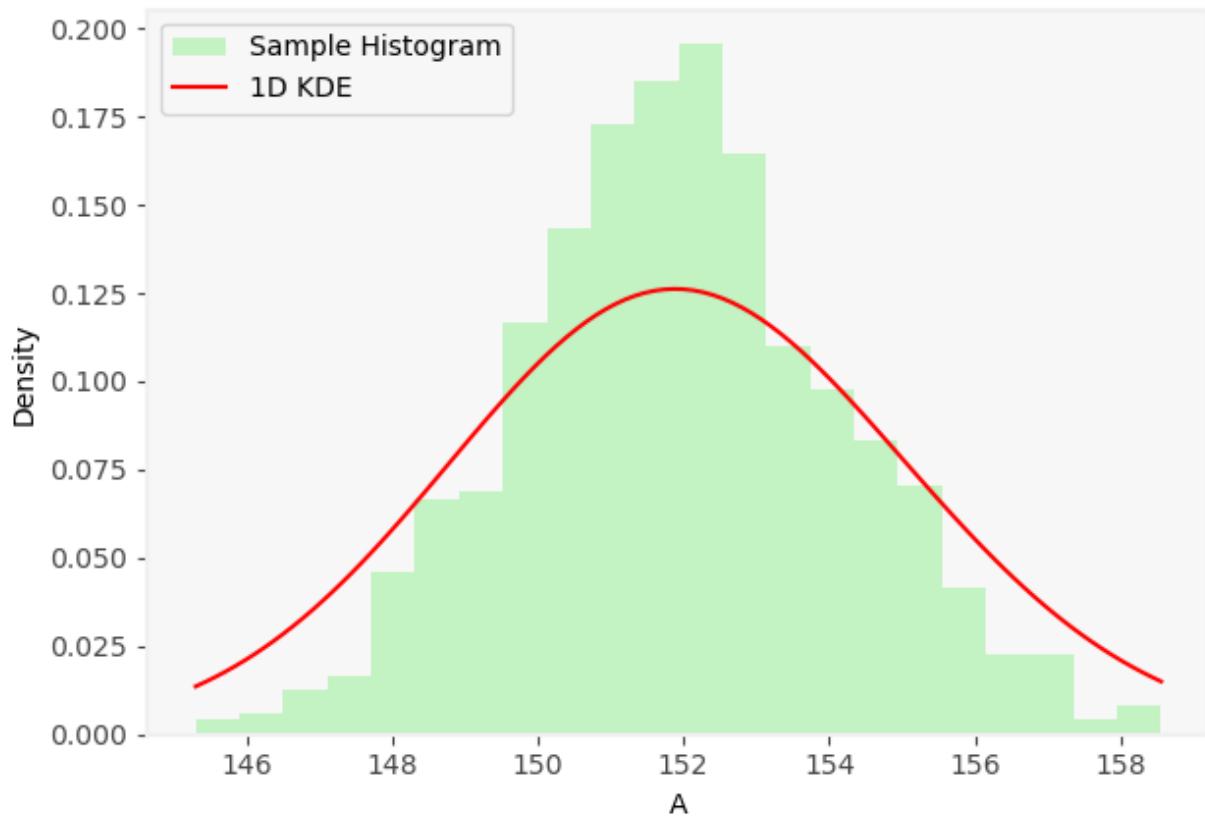
On-the-fly 2 Method, 1-D KDE for A
(iteration 15), Sample Mean: 152.8006, Sample Std: 2.6112



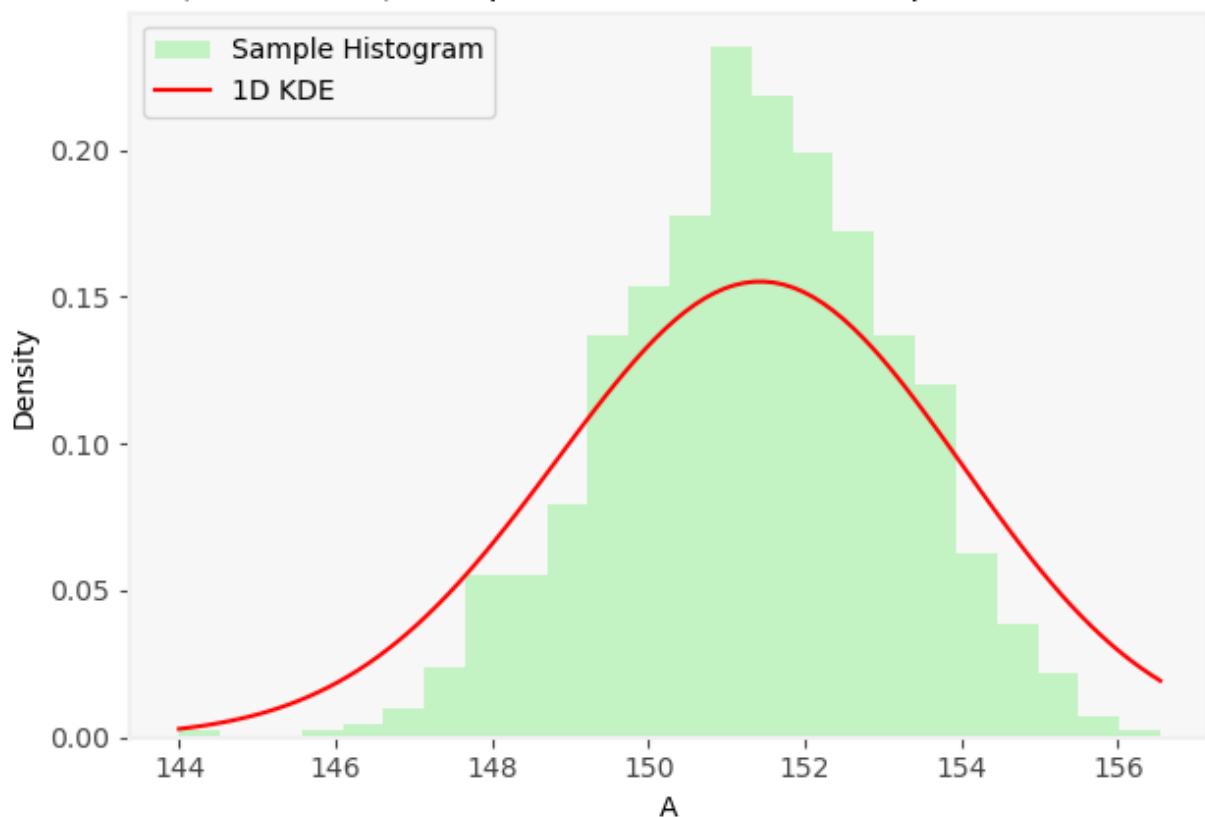
On-the-fly 2 Method, 1-D KDE for A
(iteration 16), Sample Mean: 152.4511, Sample Std: 2.1929



On-the-fly 2 Method, 1-D KDE for A
(iteration 17), Sample Mean: 151.9639, Sample Std: 2.2403



On-the-fly 2 Method, 1-D KDE for A
(iteration 18), Sample Mean: 151.3382, Sample Std: 1.8125



Playground

In []:

In []:

In []:

In []: