In [13]:

```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import seaborn as sns
%matplotlib inline
#plt.style.use('seaborn')
```

In [14]:

```
sns.set_style("ticks")
```

Preparing the dataframes

In [15]:

```
df_all = pd.read_csv('dataport-export_gas_oct2015-mar2016.csv')
len(df_all)
```

Out[15]:

```
1584823
```

In [16]:

```
df_all = df_all.set_index(pd.to_datetime(df_all['localminute']))
```

In [17]:

```
display(df_all.head(), df_all.tail())
```

localminute	localminute	dataid	meter_value
2015-10-01 00:00:10-05:00	2015-10-01 00:00:10-05	739	88858
2015-10-01 00:00:13-05:00	2015-10-01 00:00:13-05	8890	197164
2015-10-01 00:00:20-05:00	2015-10-01 00:00:20-05	6910	179118
2015-10-01 00:00:22-05:00	2015-10-01 00:00:22-05	3635	151318
2015-10-01 00:00:22-05:00	2015-10-01 00:00:22-05	1507	390354

localminute	localminute	dataid	meter_value
2016-03-31 23:59:14.336743-05:00	2016-03-31 23:59:14.336743-05	2129	201726
2016-03-31 23:59:17.427165-05:00	2016-03-31 23:59:17.427165-05	2945	161232
2016-03-31 23:59:35.370782-05:00	2016-03-31 23:59:35.370782-05	9729	138146
2016-03-31 23:59:47.816286-05:00	2016-03-31 23:59:47.816286-05	5129	166488
2016-03-31 23:59:58.923080-05:00	2016-03-31 23:59:58.92308-05	484	114174

In [18]:

```
df_all = df_all.drop(columns='localminute')
display(df_all.head())
```

localminute	dataid	meter_value
2015-10-01 00:00:10-05:00	739	88858
2015-10-01 00:00:13-05:00	8890	197164
2015-10-01 00:00:20-05:00	6910	179118
2015-10-01 00:00:22-05:00	3635	151318
2015-10-01 00:00:22-05:00	1507	390354

In [54]:

```
groups = df_all.groupby('dataid')
keys = groups.groups.keys()  # keys: an iterable of dataids or meter ids

# check if each group (grouped by meter id) is sorted in ascending order by datetime.
for key in keys:
    df_i = groups.get_group(key)
    if df_i.index.is_monotonic_increasing is False:
        print(key)

print(len(keys))
```

```
157
```

Check meterids

In [20]:

```
keys_list = list(keys)
print(keys_list)
```

```
[35, 44, 77, 94, 114, 187, 222, 252, 370, 483, 484, 661, 739, 744, 871, 1042,
1086, 1103, 1185, 1283, 1403, 1415, 1507, 1556, 1589, 1619, 1697, 1714, 1718,
1790, 1791, 1792, 1800, 1801, 2018, 2034, 2072, 2094, 2129, 2233, 2335, 2378,
2449, 2461, 2470, 2575, 2638, 2645, 2755, 2814, 2818, 2945, 2946, 2965, 2980,
3036, 3039, 3134, 3310, 3367, 3527, 3544, 3577, 3635, 3723, 3778, 3849, 3893,
3918, 4029, 4031, 4193, 4228, 4296, 4352, 4356, 4373, 4421, 4447, 4514, 4671,
4732, 4767, 4874, 4998, 5129, 5131, 5193, 5275, 5317, 5395, 5403, 5439, 5484,
5545, 5636, 5658, 5785, 5810, 5814, 5892, 5972, 6101, 6412, 6505, 6578, 6673,
6685, 6830, 6836, 6863, 6910, 7016, 7017, 7030, 7117, 7287, 7429, 7460, 7566,
7674, 7682, 7739, 7741, 7794, 7900, 7919, 7965, 7989, 8059, 8084, 8086, 8155,
8156, 8244, 8386, 8467, 8703, 8829, 8890, 8967, 9052, 9121, 9134, 9160, 9278,
9295, 9474, 9600, 9620, 9631, 9639, 9729, 9766, 9849, 9956, 9982]
```

Check data count per meter

In [21]:

```
count_list = []
for key in keys_list:
    df_i = groups.get_group(key)
    count_list.append(len(df_i.index))

print(count_list)
```

```
[11872, 1549, 10683, 36335, 2597, 914, 2731, 16774, 3641, 27628, 44034, 3622,
31430, 6058, 35070, 3830, 30029, 696, 18456, 12228, 202, 930, 32603, 3690, 26
352, 2983, 4690, 32933, 24470, 13344, 11060, 1646, 5590, 15892, 7341, 75991,
13519, 17311, 13787, 2271, 8910, 2814, 5449, 12806, 1453, 2080, 5698, 74, 68,
37, 732, 9895, 45, 5017, 3225, 336, 5400, 4017, 10200, 12068, 10853, 2221, 36
74, 9186, 8141, 13609, 1563, 26844, 6325, 10356, 12534, 1019, 799, 1176, 330
4, 1924, 1692, 3269, 9158, 19074, 21, 5303, 7583, 2, 13974, 4486, 15187, 1946
4, 2289, 1039, 1545, 25559, 5972, 2056, 33, 3411, 493, 12103, 42234, 42424, 1
4139, 5243, 3, 15783, 1862, 928, 10694, 78, 2389, 4520, 672, 69349, 2929, 252
79, 17915, 20493, 41005, 13212, 3646, 32, 29329, 3391, 4433, 5644, 8529, 228
0, 2893, 641, 39723, 529, 4686, 4391, 5423, 25296, 919, 603, 9020, 258, 1488
3, 16574, 6695, 5933, 4510, 14064, 72, 6145, 26534, 3473, 330, 23, 4411, 1379
6, 12361, 2282, 2741, 1292, 1540]
```

Check culmulative reading decreases (malfunctions)

In [22]:

```
error_count_list = []
for key in keys_list:
    df_i = groups.get_group(key)
    prev_meter_value = 0
    error_count = 0
    for i in range(len(df_i.index)):
        curr_meter_value = df_i.iloc[i][1]
        if i is 0:
            prev_meter_value = curr_meter_value
        else:
            if curr_meter_value < prev_meter_value:
                error_count += 1
            prev_meter_value = curr_meter_value
    error_count_list.append(error_count)

print(error_count_list)
```

```
[1, 0, 1, 6, 0, 0, 0, 0, 0, 1, 9, 0, 0, 0, 0, 1, 1, 0, 135, 0, 0, 0, 2, 12,
0, 0, 0, 0, 4, 1, 0, 0, 0, 1, 0, 0, 0, 0, 3, 0, 5, 0, 93, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 18, 0, 0, 1, 18, 0, 0, 0, 0, 0, 2, 0, 0, 16, 1, 0, 0,
0, 0, 0, 0, 0, 141, 0, 0, 0, 0, 1, 76, 1, 4, 0, 0, 0, 156, 0, 0, 0, 0, 0, 0,
10, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 51, 0, 0, 0, 1, 90, 123, 0, 0, 0, 0, 0, 0,
1, 0, 1, 0, 0, 0, 2, 0, 0, 0, 0, 151, 0, 0, 0, 0, 0, 44, 0, 0, 0, 115, 0, 0,
0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 2]
```

In [23]:

```
percent_error_list = []
for i in range(157):
    percent_error = round(error_count_list[i] / count_list[i] * 100, 6)
    percent_error_list.append(percent_error)

print(percent_error_list)
```

```
[0.008423, 0.0, 0.009361, 0.016513, 0.0, 0.0, 0.0, 0.0, 0.0, 0.00362, 0.02043
9, 0.0, 0.0, 0.0, 0.0, 0.02611, 0.00333, 0.0, 0.731469, 0.0, 0.0, 0.0, 0.0061
34, 0.325203, 0.0, 0.0, 0.0, 0.0, 0.016347, 0.007494, 0.0, 0.0, 0.0, 0.00629
2, 0.0, 0.0, 0.0, 0.0, 0.02176, 0.0, 0.056117, 0.0, 1.706735, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.448096, 0.0, 0.0, 0.
009214, 0.810446, 0.0, 0.0, 0.0, 0.0, 0.0, 0.00745, 0.0, 0.0, 0.127653, 0.098
135, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.739226, 0.0, 0.0, 0.0, 0.0, 0.00715
6, 1.69416, 0.006585, 0.020551, 0.0, 0.0, 0.0, 0.610353, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.023678, 0.002357, 0.007073, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.
0, 1.128319, 0.0, 0.0, 0.0, 0.003956, 0.502372, 0.600205, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.022558, 0.0, 0.011725, 0.0, 0.0, 0.0, 0.005035, 0.0, 0.0, 0.0, 0.
0, 0.596932, 0.0, 0.0, 0.0, 0.0, 0.0, 0.265476, 0.0, 0.0, 0.0, 0.817691, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.014497, 0.0, 0.0, 0.0, 0.0, 0.12987]
```

In [24]:

```
df_error_metric = pd.DataFrame(keys_list, columns=['meterid'])
df_error_metric = df_error_metric.assign(count=count_list)
df_error_metric = df_error_metric.assign(errors=error_count_list)
df_error_metric = df_error_metric.assign(percentage=percent_error_list)
df_error_metric
```

Out[24]:

	meterid	count	errors	percentage
0	35	11872	1	0.008423
1	44	1549	0	0.000000
2	77	10683	1	0.009361
3	94	36335	6	0.016513
4	114	2597	0	0.000000
5	187	914	0	0.000000
6	222	2731	0	0.000000
7	252	16774	0	0.000000
8	370	3641	0	0.000000
9	483	27628	1	0.003620
10	484	44034	9	0.020439
11	661	3622	0	0.000000
12	739	31430	0	0.000000
13	744	6058	0	0.000000
14	871	35070	0	0.000000
15	1042	3830	1	0.026110
16	1086	30029	1	0.003330
17	1103	696	0	0.000000
18	1185	18456	135	0.731469
19	1283	12228	0	0.000000
20	1403	202	0	0.000000
21	1415	930	0	0.000000
22	1507	32603	2	0.006134
23	1556	3690	12	0.325203
24	1589	26352	0	0.000000
25	1619	2983	0	0.000000
26	1697	4690	0	0.000000
27	1714	32933	0	0.000000
28	1718	24470	4	0.016347
29	1790	13344	1	0.007494
...	...	...	...	...
127	7965	641	0	0.000000
128	7989	39723	2	0.005035
129	8059	529	0	0.000000

	meterid	count	errors	percentage
130	8084	4686	0	0.000000
131	8086	4391	0	0.000000
132	8155	5423	0	0.000000
133	8156	25296	151	0.596932
134	8244	919	0	0.000000
135	8386	603	0	0.000000
136	8467	9020	0	0.000000
137	8703	258	0	0.000000
138	8829	14883	0	0.000000
139	8890	16574	44	0.265476
140	8967	6695	0	0.000000
141	9052	5933	0	0.000000
142	9121	4510	0	0.000000
143	9134	14064	115	0.817691
144	9160	72	0	0.000000
145	9278	6145	0	0.000000
146	9295	26534	0	0.000000
147	9474	3473	0	0.000000
148	9600	330	0	0.000000
149	9620	23	0	0.000000
150	9631	4411	0	0.000000
151	9639	13796	2	0.014497
152	9729	12361	0	0.000000
153	9766	2282	0	0.000000
154	9849	2741	0	0.000000
155	9956	1292	0	0.000000
156	9982	1540	2	0.129870

157 rows × 4 columns

In [25]:

```
df_error_metric.sort_values(by=['count'])
```

Out[25]:

	meterid	count	errors	percentage
83	4874	2	0	0.000000
102	6101	3	0	0.000000
80	4671	21	0	0.000000
149	9620	23	0	0.000000
119	7566	32	0	0.000000
94	5545	33	0	0.000000
49	2814	37	0	0.000000
52	2946	45	0	0.000000
48	2755	68	0	0.000000
144	9160	72	0	0.000000
47	2645	74	0	0.000000
107	6685	78	0	0.000000
20	1403	202	0	0.000000
137	8703	258	0	0.000000
148	9600	330	0	0.000000
55	3036	336	0	0.000000
96	5658	493	0	0.000000
129	8059	529	0	0.000000
135	8386	603	0	0.000000
127	7965	641	0	0.000000
110	6863	672	0	0.000000
17	1103	696	0	0.000000
50	2818	732	0	0.000000
72	4228	799	0	0.000000
5	187	914	0	0.000000
134	8244	919	0	0.000000
105	6578	928	0	0.000000
21	1415	930	0	0.000000
71	4193	1019	1	0.098135
89	5317	1039	0	0.000000
...	...	...	...	...
139	8890	16574	44	0.265476
7	252	16774	0	0.000000
37	2094	17311	0	0.000000

	meterid	count	errors	percentage
114	7030	17915	90	0.502372
18	1185	18456	135	0.731469
79	4514	19074	141	0.739226
87	5193	19464	4	0.020551
115	7117	20493	123	0.600205
28	1718	24470	4	0.016347
113	7017	25279	1	0.003956
133	8156	25296	151	0.596932
91	5403	25559	156	0.610353
24	1589	26352	0	0.000000
146	9295	26534	0	0.000000
67	3893	26844	2	0.007450
9	483	27628	1	0.003620
120	7674	29329	0	0.000000
16	1086	30029	1	0.003330
12	739	31430	0	0.000000
22	1507	32603	2	0.006134
27	1714	32933	0	0.000000
14	871	35070	0	0.000000
3	94	36335	6	0.016513
128	7989	39723	2	0.005035
116	7287	41005	0	0.000000
98	5810	42234	10	0.023678
99	5814	42424	1	0.002357
10	484	44034	9	0.020439
111	6910	69349	0	0.000000
35	2034	75991	0	0.000000

157 rows × 4 columns

In [26]:

```
df_error_metric.sort_values(by=['percentage'], ascending=False)
```

Out[26]:

	meterid	count	errors	percentage
42	2449	5449	93	1.706735
85	5129	4486	76	1.694160
109	6836	4520	51	1.128319
143	9134	14064	115	0.817691
61	3544	2221	18	0.810446
79	4514	19074	141	0.739226
18	1185	18456	135	0.731469
91	5403	25559	156	0.610353
115	7117	20493	123	0.600205
133	8156	25296	151	0.596932
114	7030	17915	90	0.502372
57	3134	4017	18	0.448096
23	1556	3690	12	0.325203
139	8890	16574	44	0.265476
156	9982	1540	2	0.129870
70	4031	12534	16	0.127653
71	4193	1019	1	0.098135
40	2335	8910	5	0.056117
15	1042	3830	1	0.026110
98	5810	42234	10	0.023678
122	7739	4433	1	0.022558
38	2129	13787	3	0.021760
87	5193	19464	4	0.020551
10	484	44034	9	0.020439
3	94	36335	6	0.016513
28	1718	24470	4	0.016347
151	9639	13796	2	0.014497
124	7794	8529	1	0.011725
2	77	10683	1	0.009361
60	3527	10853	1	0.009214
...	...	...	...	...
63	3635	9186	0	0.000000
96	5658	493	0	0.000000
80	4671	21	0	0.000000

	meterid	count	errors	percentage
95	5636	3411	0	0.000000
32	1800	5590	0	0.000000
93	5484	2056	0	0.000000
92	5439	5972	0	0.000000
90	5395	1545	0	0.000000
89	5317	1039	0	0.000000
88	5275	2289	0	0.000000
17	1103	696	0	0.000000
19	1283	12228	0	0.000000
83	4874	2	0	0.000000
82	4767	7583	0	0.000000
81	4732	5303	0	0.000000
20	1403	202	0	0.000000
64	3723	8141	0	0.000000
1	44	1549	0	0.000000
77	4421	3269	0	0.000000
76	4373	1692	0	0.000000
75	4356	1924	0	0.000000
74	4352	3304	0	0.000000
73	4296	1176	0	0.000000
72	4228	799	0	0.000000
21	1415	930	0	0.000000
69	4029	10356	0	0.000000
68	3918	6325	0	0.000000
66	3849	1563	0	0.000000
65	3778	13609	0	0.000000
78	4447	9158	0	0.000000

157 rows × 4 columns

In [31]:

```
df_error_metric_sorted = df_error_metric.sort_values(by=['percentage'], ascending=False)
keys_list2 = df_error_metric_sorted["meterid"].tolist()
keys_list2 = keys_list2[18:]
```

Checking culmulative reading sharp increases (potential malfunction)

In [55]:

```
max_increment_list = []

for key in keys_list2:
    df_i = groups.get_group(key)
    prev_meter_value = 0
    max_increment = 0
    prev_time = df_i.index[0]
    for i in range(len(df_i.index)):
        curr_meter_value = df_i.iloc[i][1]
        curr_time = df_i.index[i]
        if i != 0:
            if curr_meter_value > prev_meter_value:
                increment = curr_meter_value - prev_meter_value
                time_period = curr_time - prev_time
                time_period_seconds = time_period.total_seconds()
                increment /= time_period_seconds
                if increment > max_increment:
                    max_increment = increment
        prev_meter_value = curr_meter_value
        prev_time = curr_time
    max_increment_list.append(max_increment)

print(max_increment_list)
```

[4.0, 33.59692924066741, 0.03418884303177361, 0.048381790384280435, 0.1481484
5541901864, 0.13333608894583823, 0.16666666666666666, 0.13114754098360656, 42
9.097886280175, 0.033898418845541206, 0.033058530442203964, 0.067799749642644
47, 0.034482758620689655, 0.06557377049180328, 0.13333333333333333, 0.0502097
60479525974, 0.0339024158963275, 0.061494807793519866, 0.15446400988569664,
0.14285714285714285, 0.47619047619047616, 0.15584415584415584, 0.133366657215
4162, 0.13336150817462702, 0.15384615384615385, 0.003846595259398303, 0.15382
138860258882, 0.03361371767751127, 0.05128159939035067, 0.002695311805944049,
0.13800028207257656, 0.03333333333333333, 0.13333496890895197, 0.008316002195
707323, 0.06666749334358413, 0.06224158732280279, 0.14285714285714285, 0.0344
83390023451806, 0.125, 0.06896551724137931, 0.03388881078459359, 0.1538484379
0373195, 0.01694908675693445, 0.035595105672969966, 0.025925925925925925, 0.0
3309383216681436, 0.014814814814814815, 0.03278688524590164, 0.02777751882957
4466, 0.034482758620689655, 0.019048486025854454, 0.011612903225806452, 0.075
6537316973882, 0.03448348336838321, 0.032653423477571476, 0.0322582221651825
2, 0.01666686384492532, 0.016165793243965013, 0.03389830508474576, 0.14285714
285714285, 0.033426062802558096, 0.01380491367886987, 0.026672067838197432,
0.03389830508474576, 0.12903115921851505, 0.03333333333333333, 0.133333333333
33333, 0.049586776859504134, 0.03361346232611546, 0.034482758620689655, 0.008
326517648434678, 0.022474298701027115, 0.050101829672475444, 0.03361402839793
793, 0.0446927374301676, 0.03448261177232573, 0.05043752408549392, 0.01328892
7849367558, 2.0, 0.009399341027970534, 0.0344838294030475, 0.0095236763678649
2, 0.03305777802078918, 0.05020957560540321, 0.03448258382966128, 0.033899378
37353901, 0.03448062617230896, 0.01820905299335929, 0.03333333333333333, 0.03
448227943590991, 0.008873993004231985, 0.03448278596910611, 0.016806705882369
75, 0.009172605209594481, 0.0190705693199396, 0.03389841539823316, 0.03389830
508474576, 0.03252000634855564, 0.03361388433593494, 0.03448428128007514, 0.1
3335022436175248, 0.06896551724137931, 0.03389867682108311, 0.033898305084745
76, 0.03448843077829645, 0.04395610819958671, 0.14305406699851397, 0.06557377
049180328, 0.1428711238171164, 0.03897053416060781, 0.01486988847583643, 0.00
05767348555060087, 0.03361403800211537, 0.03448302259417296, 0.03389755014608
658, 0.033613849026808904, 0.03389856018577496, 0.011764547937529945, 0.03448
295362773776, 0.026644408904429567, 0.049383112740066126, 0.00247820027627263
53, 0.03452477586817598, 0.03389830508474576, 0.0076921244866119875, 0.033898
724509642236, 0.01834862385321101, 0.01694915254237288, 0.028223464229975924,
0.03333379556196513, 0.06666666666666667, 0.03333333333333333, 0.033898305084
74576, 0.033333165556400035, 0.1290305099092206, 0.125, 0.02253586677566289,
0.13333333333333333, 24.1579100133673]

In [56]:

```
df_max_increase = pd.DataFrame(keys_list2, columns=['meterid'])
df_max_increase = df_max_increase.assign(maxIncrease=max_increment_list)
df_max_increase.sort_values(by=['maxIncrease'], ascending=False)
```

Out[56]:

	meterid	maxIncrease
8	9639	429.097886
1	5810	33.596929
138	4447	24.157910
0	1042	4.000000
78	7919	2.000000
20	7989	0.476190
6	94	0.166667
21	7017	0.155844
18	1801	0.154464
41	6910	0.153848
24	5814	0.153846
26	6673	0.153821
4	5193	0.148148
106	2034	0.143054
108	1589	0.142871
19	1507	0.142857
36	7287	0.142857
59	9295	0.142857
30	6412	0.138000
22	483	0.133367
23	1086	0.133362
100	1714	0.133350
5	484	0.133336
32	7674	0.133335
66	222	0.133333
14	3893	0.133333
137	3778	0.133333
7	1718	0.131148
64	871	0.129031
134	4029	0.129031
...	...	...
119	1103	0.026644
44	6578	0.025926
136	3849	0.022536

	meterid	maxIncrease
71	8244	0.022474
94	2645	0.019071
50	9956	0.019048
126	44	0.018349
87	3036	0.018209
127	4421	0.016949
42	370	0.016949
92	2818	0.016807
56	9620	0.016667
57	9600	0.016166
110	5658	0.014870
46	6863	0.014815
61	9160	0.013805
77	7965	0.013289
117	5317	0.011765
51	9849	0.011613
81	2755	0.009524
79	5545	0.009399
93	2814	0.009173
90	2946	0.008874
70	8386	0.008327
33	7566	0.008316
124	1403	0.007692
25	6685	0.003847
29	6101	0.002695
121	4874	0.002478
111	4671	0.000577

139 rows × 2 columns

In [57]:

```
max_increment_list.sort(reverse=True)
print(max_increment_list)
avg_max_increment_list = []

for i in range(138):
    sublist = max_increment_list[i:]
    avg_max_increment_list.append(round(sum(sublist) / len(sublist), 3))

print(avg_max_increment_list)
```

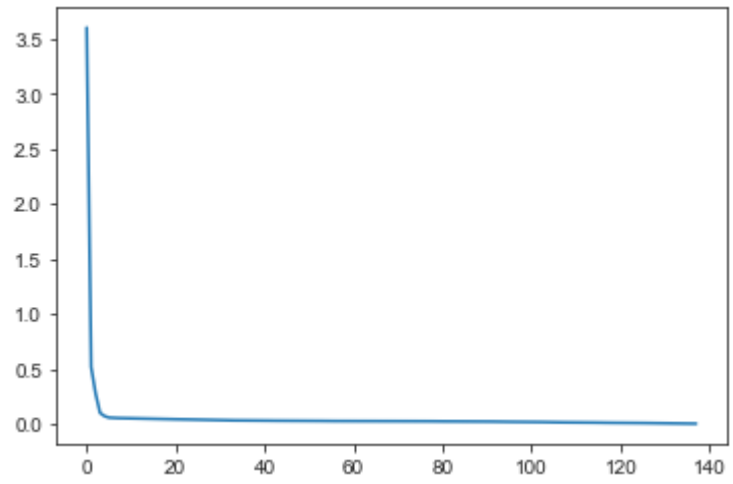
[429.097886280175, 33.59692924066741, 24.1579100133673, 4.0, 2.0, 0.476190476
19047616, 0.16666666666666666, 0.15584415584415584, 0.15446400988569664, 0.15
384843790373195, 0.15384615384615385, 0.15382138860258882, 0.1481484554190186
4, 0.14305406699851397, 0.1428711238171164, 0.14285714285714285, 0.1428571428
5714285, 0.14285714285714285, 0.13800028207257656, 0.1333666572154162, 0.1333
6150817462702, 0.13335022436175248, 0.13333608894583823, 0.13333496890895197,
0.13333333333333333, 0.13333333333333333, 0.13333333333333333, 0.131147540983
60656, 0.12903115921851505, 0.1290305099092206, 0.125, 0.125, 0.0756537316973
882, 0.06896551724137931, 0.06896551724137931, 0.06779974964264447, 0.0666674
9334358413, 0.06666666666666667, 0.06557377049180328, 0.06557377049180328, 0.
06224158732280279, 0.061494807793519866, 0.05128159939035067, 0.0504375240854
9392, 0.050209760479525974, 0.05020957560540321, 0.050101829672475444, 0.0495
86776859504134, 0.049383112740066126, 0.048381790384280435, 0.044692737430167
6, 0.04395610819958671, 0.03897053416060781, 0.035595105672969966, 0.03452477
586817598, 0.03448843077829645, 0.03448428128007514, 0.0344838294030475, 0.03
448348336838321, 0.034483390023451806, 0.03448302259417296, 0.034482953627737
76, 0.03448278596910611, 0.034482758620689655, 0.034482758620689655, 0.034482
758620689655, 0.03448261177232573, 0.03448258382966128, 0.03448227943590991,
0.03448062617230896, 0.03418884303177361, 0.0339024158963275, 0.0338993783735
3901, 0.033898724509642236, 0.03389867682108311, 0.03389856018577496, 0.03389
8418845541206, 0.03389841539823316, 0.03389830508474576, 0.03389830508474576,
0.03389830508474576, 0.03389830508474576, 0.03389830508474576, 0.033898305084
74576, 0.03389755014608658, 0.03388881078459359, 0.03361403800211537, 0.03361
402839793793, 0.03361388433593494, 0.033613849026808904, 0.03361371767751127,
0.03361346232611546, 0.033426062802558096, 0.03333379556196513, 0.03333333333
333333, 0.03333333333333333, 0.03333333333333333, 0.03333333333333333, 0.0333
33165556400035, 0.03309383216681436, 0.033058530442203964, 0.0330577780207891
8, 0.03278688524590164, 0.032653423477571476, 0.03252000634855564, 0.03225822
216518252, 0.028223464229975924, 0.027777518829574466, 0.026672067838197432,
0.026644408904429567, 0.025925925925925925, 0.02253586677566289, 0.0224742987
01027115, 0.0190705693199396, 0.019048486025854454, 0.01834862385321101, 0.01
820905299335929, 0.01694915254237288, 0.01694908675693445, 0.01680670588236975
, 0.01666686384492532, 0.016165793243965013, 0.01486988847583643, 0.01481481
4814814815, 0.01380491367886987, 0.013288927849367558, 0.011764547937529945,
0.011612903225806452, 0.00952367636786492, 0.009399341027970534, 0.0091726052
09594481, 0.008873993004231985, 0.008326517648434678, 0.008316002195707323,
0.0076921244866119875, 0.00384659525939830 3, 0.002695311805944049, 0.00247820
02762726353, 0.0005767348555060087]
[3.6, 0.517, 0.276, 0.1, 0.071, 0.057, 0.054, 0.053, 0.052, 0.051, 0.05, 0.0
5, 0.049, 0.048, 0.047, 0.046, 0.046, 0.045, 0.044, 0.043, 0.042, 0.042, 0.04
1, 0.04, 0.039, 0.038, 0.038, 0.037, 0.036, 0.035, 0.034, 0.033, 0.033, 0.03
2, 0.032, 0.031, 0.031, 0.031, 0.03, 0.03, 0.03, 0.029, 0.029, 0.029, 0.029,
0.028, 0.028, 0.028, 0.028, 0.027, 0.027, 0.027, 0.027, 0.027, 0.026, 0.026,
0.026, 0.026, 0.026, 0.026, 0.026, 0.026, 0.026, 0.026, 0.025, 0.025, 0.025,
0.025, 0.025, 0.025, 0.025, 0.024, 0.024, 0.024, 0.024, 0.024, 0.024, 0.024,
0.023, 0.023, 0.023, 0.023, 0.023, 0.022, 0.022, 0.022, 0.022, 0.022, 0.021,
0.021, 0.021, 0.021, 0.02, 0.02, 0.02, 0.019, 0.019, 0.019, 0.018, 0.018, 0.0
18, 0.017, 0.017, 0.016, 0.016, 0.015, 0.015, 0.014, 0.014, 0.014, 0.013, 0.0
13, 0.012, 0.012, 0.012, 0.011, 0.011, 0.011, 0.01, 0.01, 0.01, 0.009, 0.009,
0.009, 0.008, 0.008, 0.007, 0.007, 0.006, 0.006, 0.006, 0.005, 0.005, 0.004,
0.003, 0.002, 0.002, 0.002]

In [58]:

```
plt.plot(range(138), avg_max_increment_list)
```

Out[58]:

```
[<matplotlib.lines.Line2D at 0x18881666358>]
```

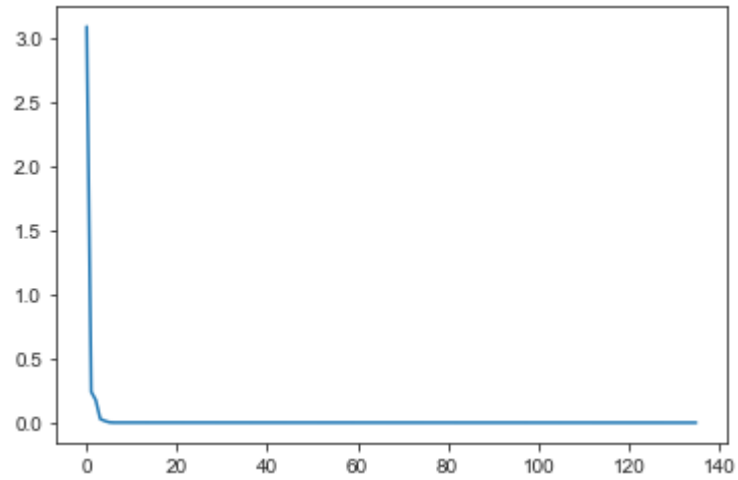


In [59]:

```
marginal_avg_max_increment_list = []

for i in range(len(avg_max_increment_list) - 1):
    if i != 0:
        marginal_avg_max_increment_list.append(avg_max_increment_list[i - 1] - avg_max_inc
rement_list[i])

plt.plot(range(len(marginal_avg_max_increment_list)), marginal_avg_max_increment_list)
```

Out[59]:

```
[<matplotlib.lines.Line2D at 0x18881d26b70>]
```

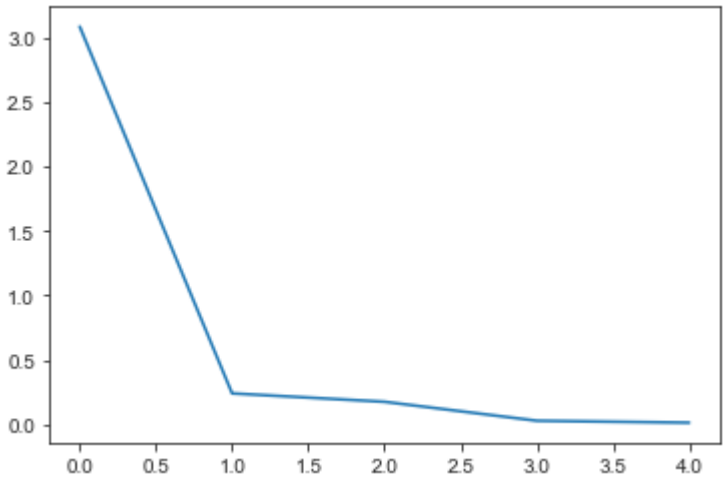


In [62]:

```
plt.plot(range(len(marginal_avg_max_increment_list[:5])), marginal_avg_max_increment_list
[:5])
```

Out[62]:

```
[<matplotlib.lines.Line2D at 0x188801b2780>]
```



In [16]:

```
average_increment_list = []

for key in keys_list2:
    df_i = groups.get_group(key)
    prev_meter_value = 0
    sum_increment = 0
    sum_count = 0
    for i in range(len(df_i.index)):
        curr_meter_value = df_i.iloc[i][1]
        if i != 0:
            if curr_meter_value > prev_meter_value:
                sum_increment += (curr_meter_value - prev_meter_value)
                sum_count += 1
        prev_meter_value = curr_meter_value
    average_increment = round(sum_increment / sum_count, 6)
    average_increment_list.append(average_increment)

print(average_increment_list)
```

```
[10.42692, 3.94857, 6.987342, 6.316891, 4.386805, 3.005447, 4.651663, 5.73517
6, 18.584844, 8.742521, 4.071329, 6.862478, 5.094871, 12.15355, 5.505476, 7.3
90728, 5.681843, 5.994563, 6.793639, 5.756627, 3.738176, 6.925804, 5.786694,
4.887492, 3.587503, 244.101695, 3.969605, 12.521146, 15.581065, 671.0, 7.6197
04, 8.667235, 4.791174, 113.666667, 20.403064, 8.870763, 4.23814, 14.485459,
3.998305, 36.7875, 21.560821, 3.328149, 6.844444, 8.909443, 17.276923, 12.910
288, 140.697674, 13.177066, 17.051656, 9.320015, 24.361656, 8.202899, 30.7327
59, 5.747209, 12.813866, 11.101031, 1698.0, 12.854962, 19.767045, 4.620885,
6.796938, 65.1875, 31.607397, 12.11974, 4.700599, 3.471452, 69.313496, 5.7715
6, 92.021622, 6.590345, 24.393204, 6.771372, 7.172503, 13.294375, 13.77413,
9.859155, 34.51634, 54.532934, 13.586796, 329.741935, 10.779982, 81.612903,
8.841008, 6.745544, 6.569719, 10.733582, 13.59612, 17.582222, 8.662338, 7.062
817, 113.952381, 4.616251, 35.869416, 218.882353, 143.827586, 11.758442, 15.7
4145, 20.8879, 17.993769, 7.651042, 5.768904, 28.029205, 34.030418, 5.840091,
4.574045, 8.610949, 3.229239, 16.232798, 8.566618, 5.823775, 26.951111, 155.8
94737, 10.470128, 11.210637, 15.312916, 8.222958, 19.85124, 3.655738, 24.7977
84, 36.617414, 5.594539, 6986.0, 13.581808, 10.065766, 15.836066, 8.663411, 2
1.403189, 10.338756, 19.900344, 19.870085, 22.982659, 24.935393, 20.062992, 3
2.483636, 14.371047, 14.70216, 22.381955, 6.792435, 6.159564]
```

In [17]:

```
delta_1percent_list = []
delta_5percent_list = []
delta_10percent_list = []
delta_20percent_list = []
delta_50percent_list = []
constant_list = []
decrease_list = []

for key in keys_list2:
    df_i = groups.get_group(key)
    prev_meter_value = 0
    delta_1percent_count = 0
    delta_5percent_count = 0
    delta_10percent_count = 0
    delta_20percent_count = 0
    delta_50percent_count = 0
    constant_count = 0
    decrease_count = 0
    average_increment = average_increment_list[keys_list2.index(key)]
    for i in range(len(df_i.index)):
        curr_meter_value = df_i.iloc[i][1]
        if i != 0:
            if curr_meter_value > prev_meter_value:
                increment = curr_meter_value - prev_meter_value
                delta_percentage = round(abs(increment - average_increment) / average_incr
ement * 100, 3)
                if delta_percentage <= 1:
                    delta_1percent_count += 1
                elif delta_percentage <= 5:
                    delta_5percent_count += 1
                elif delta_percentage <= 10:
                    delta_10percent_count += 1
                elif delta_percentage <= 20:
                    delta_20percent_count += 1
                else:
                    delta_50percent_count += 1
            elif curr_meter_value == prev_meter_value:
                constant_count += 1
            else:
                decrease_count += 1

        prev_meter_value = curr_meter_value

    delta_1percent_list.append(delta_1percent_count)
    delta_5percent_list.append(delta_5percent_count)
    delta_10percent_list.append(delta_10percent_count)
    delta_20percent_list.append(delta_20percent_count)
    delta_50percent_list.append(delta_50percent_count)
    constant_list.append(constant_count)
    decrease_list.append(decrease_count)
```

In [18]:

```
df_increase_error = pd.DataFrame(keys_list2, columns=['meterid'])
df_increase_error = df_increase_error.assign(averageIncrement=average_increment_list)
df_increase_error = df_increase_error.assign(within1percent=delta_1percent_list)
df_increase_error = df_increase_error.assign(within5percent=delta_5percent_list)
df_increase_error = df_increase_error.assign(within10percent=delta_10percent_list)
df_increase_error = df_increase_error.assign(within20percent=delta_20percent_list)
df_increase_error = df_increase_error.assign(within50percent=delta_50percent_list)
df_increase_error = df_increase_error.assign(constant=constant_list)
df_increase_error = df_increase_error.assign(decrease=decrease_list)
df_increase_error
```

Out[18]:

	meterid	averageIncrement	within1percent	within5percent	within10percent	within20percent	v
0	1042	10.426920	0	44	0	37	
1	5810	3.948570	0	881	0	0	
2	7739	6.987342	0	0	0	276	
3	2129	6.316891	0	0	691	0	
4	5193	4.386805	0	0	688	0	
5	484	3.005447	0	0	0	0	
6	94	4.651663	0	0	0	1069	
7	1718	5.735176	0	345	0	0	
8	9639	18.584844	0	97	64	228	
9	7794	8.742521	0	0	265	176	
10	77	4.071329	0	268	0	0	
11	3527	6.862478	0	0	0	541	
12	35	5.094871	0	0	0	249	
13	1790	12.153550	0	293	0	489	
14	3893	5.505476	0	0	286	0	
15	4998	7.390728	0	0	269	411	
16	5892	5.681843	0	0	262	0	
17	5131	5.994563	362	0	0	0	
18	1801	6.793639	0	0	0	303	
19	1507	5.756627	0	672	0	0	
20	7989	3.738176	0	0	760	0	
21	7017	6.925804	0	0	0	1013	
22	483	5.786694	0	573	0	0	
23	1086	4.887492	0	0	0	760	
24	5814	3.587503	0	0	0	543	
25	6685	244.101695	0	0	1	3	
26	6673	3.969605	271	0	0	0	
27	7741	12.521146	0	45	0	51	
28	5972	15.581065	0	83	0	230	
29	6101	671.000000	0	0	0	0	
...	...	...	...	...	...	...	
109	3635	5.823775	0	250	0	0	
110	5658	26.951111	0	8	0	7	
111	4671	155.894737	0	0	1	0	

	meterid	averageIncrement	within1percent	within5percent	within10percent	within20percent	v
112	5636	10.470128	0	129	0	90	
113	1800	11.210637	0	0	27	38	
114	5484	15.312916	0	43	46	45	
115	5439	8.222958	0	143	0	0	
116	5395	19.851240	22	0	24	54	
117	5317	3.655738	0	0	20	0	
118	5275	24.797784	0	30	0	43	
119	1103	36.617414	0	5	8	12	
120	1283	5.594539	0	0	413	0	
121	4874	6986.000000	1	0	0	0	
122	4767	13.581808	0	51	0	95	
123	4732	10.065766	81	0	0	68	
124	1403	15.836066	0	2	0	0	
125	3723	8.663411	0	0	221	181	
126	44	21.403189	0	6	9	10	
127	4421	10.338756	0	53	0	35	
128	4373	19.900344	32	0	26	55	
129	4356	19.870085	52	0	57	84	
130	4352	22.982659	0	22	0	13	
131	4296	24.935393	0	42	0	42	
132	4228	20.062992	14	0	8	27	
133	1415	32.483636	0	17	6	42	
134	4029	14.371047	0	107	0	183	
135	3918	14.702160	0	97	85	167	
136	3849	22.381955	0	19	13	61	
137	3778	6.792435	0	0	0	492	
138	4447	6.159564	0	138	0	0	

139 rows × 9 columns

In [19]:

```
df_increase_error.sort_values(by=['averageIncrement'], ascending=False)
```

Out[19]:

	meterid	averageIncrement	within1percent	within5percent	within10percent	within20percent	v
121	4874	6986.000000	1	0	0	0	
56	9620	1698.000000	0	0	0	0	
29	6101	671.000000	0	0	0	0	
79	5545	329.741935	0	2	1	0	
25	6685	244.101695	0	0	1	3	
93	2814	218.882353	0	0	0	2	
111	4671	155.894737	0	0	1	0	
94	2645	143.827586	0	0	0	1	
46	6863	140.697674	0	1	1	6	
90	2946	113.952381	0	1	1	5	
33	7566	113.666667	0	0	2	1	
68	8703	92.021622	1	1	1	5	
81	2755	81.612903	1	0	0	1	
66	222	69.313496	5	3	4	13	
61	9160	65.187500	0	2	1	1	
77	7965	54.532934	1	3	1	9	
39	6505	36.787500	0	6	9	18	
119	1103	36.617414	0	5	8	12	
92	2818	35.869416	8	0	9	20	
76	8059	34.516340	0	4	3	20	
102	2233	34.030418	5	0	14	27	
133	1415	32.483636	0	17	6	42	
62	187	31.607397	0	6	18	32	
52	9766	30.732759	0	6	5	11	
101	2378	28.029205	29	0	66	87	
110	5658	26.951111	0	8	0	7	
131	4296	24.935393	0	42	0	42	
118	5275	24.797784	0	30	0	43	
70	8386	24.393204	0	8	15	16	
50	9956	24.361656	0	7	6	15	
...	...	...	...	...	...	...	
22	483	5.786694	0	573	0	0	
67	8829	5.771560	0	365	0	0	
100	1714	5.768904	0	435	0	0	

	meterid	averageIncrement	within1percent	within5percent	within10percent	within20percent	v
19	1507	5.756627	0	672	0	0	
53	9729	5.747209	0	290	0	0	
7	1718	5.735176	0	345	0	0	
16	5892	5.681843	0	0	262	0	
120	1283	5.594539	0	0	413	0	
14	3893	5.505476	0	0	286	0	
12	35	5.094871	0	0	0	249	
23	1086	4.887492	0	0	0	760	
32	7674	4.791174	0	0	0	1099	
64	871	4.700599	0	0	0	726	
6	94	4.651663	0	0	0	1069	
59	9295	4.620885	0	0	0	1130	
91	2945	4.616251	0	0	0	530	
104	2094	4.574045	0	0	0	869	
4	5193	4.386805	0	0	688	0	
36	7287	4.238140	0	0	833	0	
10	77	4.071329	0	268	0	0	
38	739	3.998305	666	0	0	0	
26	6673	3.969605	271	0	0	0	
1	5810	3.948570	0	881	0	0	
20	7989	3.738176	0	0	760	0	
117	5317	3.655738	0	0	20	0	
24	5814	3.587503	0	0	0	543	
65	8967	3.471452	0	0	0	19	
41	6910	3.328149	0	0	0	0	
106	2034	3.229239	0	0	0	0	
5	484	3.005447	0	0	0	0	

139 rows × 9 columns

In [38]:

```
increase_list = average_increment_list
increase_list.sort(reverse=True)
print(increase_list)
print(len(increase_list))
```

```
[6986.0, 1698.0, 671.0, 329.741935, 244.101695, 218.882353, 155.894737, 143.8
27586, 140.697674, 113.952381, 113.666667, 92.021622, 81.612903, 69.313496, 6
5.1875, 54.532934, 36.7875, 36.617414, 35.869416, 34.51634, 34.030418, 32.483
636, 31.607397, 30.732759, 28.029205, 26.951111, 24.935393, 24.797784, 24.393
204, 24.361656, 22.982659, 22.381955, 21.560821, 21.403189, 20.8879, 20.40306
4, 20.062992, 19.900344, 19.870085, 19.85124, 19.767045, 18.584844, 17.99376
9, 17.582222, 17.276923, 17.051656, 16.232798, 15.836066, 15.74145, 15.58106
5, 15.312916, 14.70216, 14.485459, 14.371047, 13.77413, 13.59612, 13.586796,
13.581808, 13.294375, 13.177066, 12.910288, 12.854962, 12.813866, 12.521146,
12.15355, 12.11974, 11.758442, 11.210637, 11.101031, 10.779982, 10.733582, 1
0.470128, 10.42692, 10.338756, 10.065766, 9.859155, 9.320015, 8.909443, 8.870
763, 8.841008, 8.742521, 8.667235, 8.663411, 8.662338, 8.610949, 8.566618, 8.
222958, 8.202899, 7.651042, 7.619704, 7.390728, 7.172503, 7.062817, 6.987342,
6.925804, 6.862478, 6.844444, 6.796938, 6.793639, 6.792435, 6.771372, 6.74554
4, 6.590345, 6.569719, 6.316891, 6.159564, 5.994563, 5.840091, 5.823775, 5.78
6694, 5.77156, 5.768904, 5.756627, 5.747209, 5.735176, 5.681843, 5.594539, 5.
505476, 5.094871, 4.887492, 4.791174, 4.700599, 4.651663, 4.620885, 4.616251,
4.574045, 4.386805, 4.23814, 4.071329, 3.998305, 3.969605, 3.94857, 3.738176,
3.655738, 3.587503, 3.471452, 3.328149, 3.229239, 3.005447]
139
```

In [130]:

```
increase_threshold_list = []

for i in range(138):
    sublist = increase_list[i:]
    increase_threshold_list.append(sum(sublist)/len(sublist))

print(increase_threshold_list)
```

```
[91.30506469784176, 41.343507195652215, 29.251124036496336, 24.5323823014705
7, 22.271570799999985, 20.61612211194028, 19.12539857142856, 18.0892672196969
6, 17.12943272519084, 16.17890779230769, 15.42097389147287, 14.65342941406250
6, 14.044231047244098, 13.50797174603175, 13.061527552000001, 12.641156806451
615, 12.300573252032521, 12.09986073770492, 11.897236330578515, 11.6974681666
66669, 11.505712941176473, 11.314825610169494, 11.133895606837608, 10.9573999
05172416, 10.785440260869567, 10.63417916666667, 10.489781539823008, 10.36080
2866071428, 10.230739972972973, 10.101990299999999, 9.97116767889908, 9.85069
0907407406, 9.733576289719622, 9.62199850943396, 9.509796695238093, 9.4003918
55769229, 9.293569796116502, 9.187987225490195, 9.081924287128711, 8.97404267
9999997, 8.864171999999996, 8.752918193877548, 8.651558134020616, 8.554243437
499997, 8.459212084210526, 8.365406648936169, 8.27200611827957, 8.18547577173
913, 8.101403351648353, 8.016513944444446, 7.931518988764047, 7.8476394772727
3, 7.768851885057474, 7.690751802325583, 7.612160094117649, 7.53880330952381
2, 7.465823590361447, 7.391177585365857, 7.3147500493827184, 7.24000473750000
3, 7.164852063291143, 7.091192628205131, 7.016338480519483, 6.94005522368421
4, 6.865640680000003, 6.794182445945949, 6.7212296027397285, 6.65126831944444
6, 6.587051859154932, 6.522566442857145, 6.460864768115944, 6.39803069117647
4, 6.3372531194029875, 6.275288469696973, 6.2127735846153875, 6.1525705781250
03, 6.093735904761908, 6.041699145161292, 5.994686950819675, 5.94675235000000
1, 5.897697169491527, 5.8486484827586205, 5.799199596491228, 5.74805296428571
4, 5.695065963636364, 5.641068129629631, 5.585869075471699, 5.53515582692307
8, 5.482847137254903, 5.43948324, 5.39498893877551, 5.353411041666667, 5.3147
06957446809, 5.276704565217392, 5.2386904, 5.20034690909091, 5.16169269767441
9, 5.121627190476191, 5.0807659512195125, 5.037944125, 4.992957179487179, 4.9
46156789473683, 4.897524702702702, 4.850501916666666, 4.801381428571427, 4.75
6807617647059, 4.714299848484849, 4.674291625, 4.636685193548387, 4.597115533
3333344, 4.556095586206896, 4.512686142857143, 4.466159555555556, 4.416526192
307693, 4.36329888, 4.306137333333334, 4.2463240434782605, 4.185041545454545
5, 4.122163714285715, 4.073528350000001, 4.030688157894738, 3.98843894444444
5, 3.946547176470589, 3.9024774375, 3.8545835999999998, 3.800178785714285, 3.
740650615384615, 3.6868044166666665, 3.6366829999999997, 3.5932184000000005,
3.5482087777777775, 3.49553425, 3.430814857142857, 3.379588, 3.324358, 3.2585
7175, 3.1876116666666667, 3.117343]
```

In [131]:

```
increase_threshold_df = pd.DataFrame(increase_threshold_list, columns=['average increase'
])
increase_threshold_df
```

Out[131]:

	average increase
0	91.305065
1	41.343507
2	29.251124
3	24.532382
4	22.271571
5	20.616122
6	19.125399
7	18.089267
8	17.129433
9	16.178908
10	15.420974
11	14.653429
12	14.044231
13	13.507972
14	13.061528
15	12.641157
16	12.300573
17	12.099861
18	11.897236
19	11.697468
20	11.505713
21	11.314826
22	11.133896
23	10.957400
24	10.785440
25	10.634179
26	10.489782
27	10.360803
28	10.230740
29	10.101990
...	...
108	4.636685
109	4.597116
110	4.556096

	average increase
111	4.512686
112	4.466160
113	4.416526
114	4.363299
115	4.306137
116	4.246324
117	4.185042
118	4.122164
119	4.073528
120	4.030688
121	3.988439
122	3.946547
123	3.902477
124	3.854584
125	3.800179
126	3.740651
127	3.686804
128	3.636683
129	3.593218
130	3.548209
131	3.495534
132	3.430815
133	3.379588
134	3.324358
135	3.258572
136	3.187612
137	3.117343

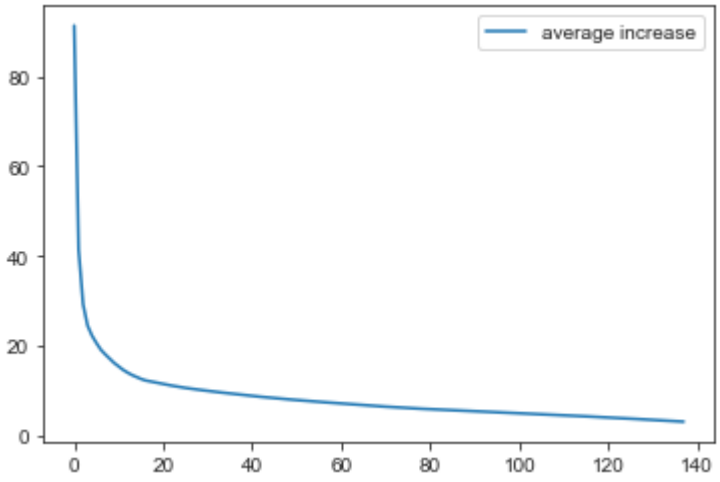
138 rows × 1 columns

In [51]:

```
sns.lineplot(data=increase_threshold_df)
```

Out[51]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x22f3ed9cb00>
```

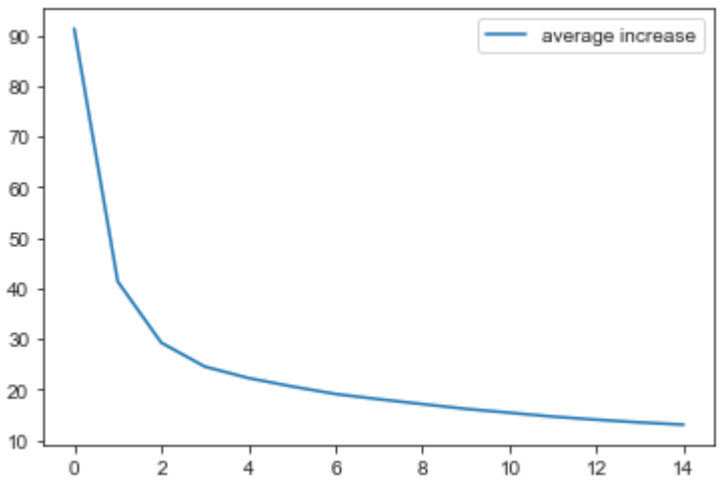


In [54]:

```
sns.lineplot(data=increase_threshold_df.iloc[:15])
```

Out[54]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x22f3f069518>
```



In [132]:

```
increase_marginal_change_list = []

for i in range(len(increase_threshold_list)):
    if i != 0:
        increase_marginal_change_list.append(increase_threshold_list[i-1] - increase_thres
hold_list[i])

print(increase_marginal_change_list)
```

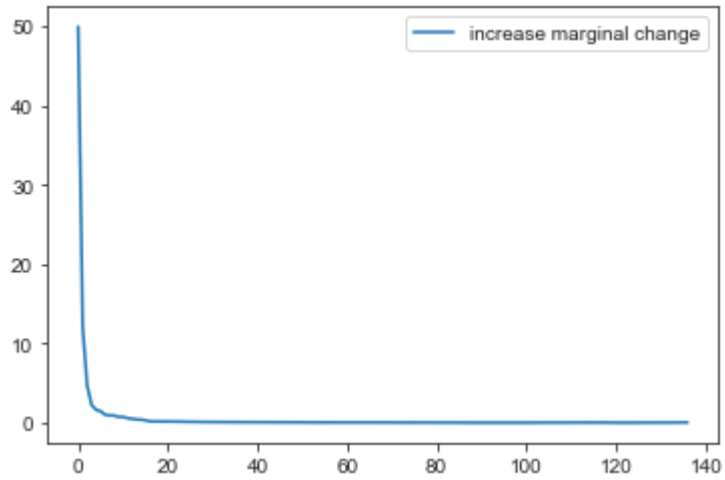
```
[49.96155750218954, 12.09238315915588, 4.718741735025766, 2.2608115014705845,
1.6554486880597032, 1.4907235405117234, 1.0361313517315978, 0.9598344945061221,
0.9505249328831482, 0.7579339008348196, 0.7675444774103646, 0.6091983668184078,
0.5362593012123487, 0.4464441940317485, 0.4203707455483858, 0.3405835544190942
7, 0.2007125143276003, 0.20262440712640561, 0.19976816391184649, 0.19175522549019597, 0.1908873310069783, 0.1809300033318859, 0.17649570166519268,
0.1719596443028486, 0.1512610942028978, 0.1443976268436611, 0.1289786737515807
9, 0.13006289309845442, 0.12874967297297424, 0.13082262110091847, 0.1204767714916741 3, 0.11711461768778442, 0.11157778028566234, 0.11220181419586694, 0.1094048394688638 1, 0.10682205965272651, 0.10558257062630716, 0.1060629383614841 7, 0.10788160712871431, 0.10987068000000022, 0.1112538061224484, 0.1013600598569315 3, 0.09731469652061975, 0.0950313532894711, 0.0938054352743567, 0.09340053065659859, 0.08653034654044056, 0.0840724200907772, 0.08488940720390659,
0.08499495568039883, 0.08387951149131734, 0.07878759221525566, 0.0781000827318989073, 0.07859170820793437, 0.07335678459383743, 0.07297971916236445, 0.0746460049955901 5, 0.07642753598313856, 0.07474531188271527, 0.0751526742088604,
0.0736594350860118, 0.07485414768564791, 0.07628325683526871, 0.0744145436842 1141, 0.07145823405405416, 0.07295284320622031, 0.0699612832952825, 0.0642164 6028951411, 0.06448541629778681, 0.06170167474120092, 0.0628340769394704, 0.0 607775717734861 94, 0.061964649706014896, 0.06251488508158509, 0.0602030064903 8468, 0.058834673363095114, 0.05203675960061549, 0.047012194341617075, 0.0479 346008196746, 0.04905518050847402, 0.04904868673290608, 0.049448886267392744,
0.05114663220551385, 0.0529870006493498, 0.05399783400673286, 0.0551990541579 3236, 0.050713248548620804, 0.052308689668175035, 0.04336389725490264, 0.0444 9430122449005, 0.04157789710884341, 0.038704084219857826, 0.0380023922294174 8, 0.038014165217391316, 0.038343490909090505, 0.038654211416490725, 0.040065 50719822766, 0.04086123925667895, 0.04282182621951236, 0.044986945512820675,
0.04680039001349634, 0.048632086770981076, 0.04702278603603638, 0.04912048809523828, 0.044573810924368296, 0.042507769162209996, 0.04000822348484867, 0.03 7606431451613354, 0.03956966021505259, 0.04101994712643808, 0.043409443349752 94, 0.046526587301586986, 0.049633363247863826, 0.05322731230769229, 0.057161 546666666396, 0.05981328985507339, 0.06128249802371499, 0.06287783116883094,
0.04863536428571358, 0.04284019210526324, 0.04224921345029298, 0.041891767973 85574, 0.04406973897058908, 0.0478938375000002, 0.05440481428571475, 0.059528 17032967017, 0.05384619871794838, 0.05012141666666681, 0.04346459999999919,
0.045009622222222934, 0.05267452777777759, 0.06471939285714301, 0.05122685714285691, 0.05522999999999989, 0.06578624999999994, 0.07096008333333348, 0.0702 6866666666676]
```

In [134]:

```
increase_marginal_change_df = pd.DataFrame(increase_marginal_change_list, columns=['increase marginal change'])
sns.lineplot(data=increase_marginal_change_df)
```

Out[134]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x22f45166f28>
```

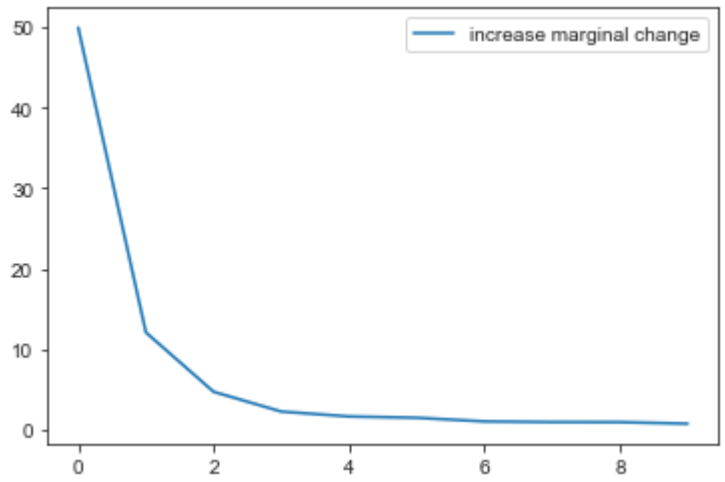


In [135]:

```
sns.lineplot(data=increase_marginal_change_df.iloc[:10])
```

Out[135]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x22f451c6cf8>
```



In [136]:

```
malfunction_sharp_increase = [4874, 9620, 6101, 5545]
malfunction_sharp_increase.sort()
print(malfunction_sharp_increase)
```

```
[4874, 5545, 6101, 9620]
```

Checking Sampling Period

Getting a list containing the average sampling period of every meter

In [128]:

```
samp_period_list = []

for key in keys_list:
    df_i = groups.get_group(key)
    start = df_i.head(1).index.item()
    end = df_i.tail(1).index.item()
    period = end - start
    period_seconds = period.total_seconds()
    sample_count = len(df_i.index)
    samp_period_list.append(round(period_seconds / sample_count, 3))

print(samp_period_list)
print(len(samp_period_list))
```

```
[1331.54, 7987.478, 1479.464, 435.013, 6075.927, 15156.239, 5788.803, 942.5,
4340.262, 572.213, 359.067, 4362.608, 503.056, 2608.427, 450.83, 4064.046, 52
6.513, 14797.415, 856.522, 1292.74, 7512.779, 14883.727, 484.959, 4283.89, 59
9.964, 4644.924, 3370.888, 480.089, 646.099, 1184.34, 1429.335, 9431.604, 282
7.491, 994.738, 2153.605, 208.063, 1169.153, 912.988, 1146.018, 6911.861, 177
3.099, 5533.95, 2900.413, 1234.015, 9514.713, 7595.324, 2758.475, 169030.934,
22635.927, 96812.003, 18158.627, 1370.849, 33554.24, 3101.576, 4899.285, 459
8.992, 2911.828, 3933.222, 1329.611, 1309.967, 1456.039, 7116.733, 4302.707,
1721.202, 1702.338, 1158.766, 10086.732, 588.913, 2142.872, 1526.569, 1261.06
7, 15374.998, 19324.431, 13435.344, 4729.024, 8192.441, 8183.171, 4807.326, 1
716.916, 828.746, 560990.309, 2975.533, 2083.478, 1409490.602, 1131.373, 352
4.266, 1041.042, 712.031, 6906.726, 1488.945, 9814.518, 618.566, 2320.96, 768
7.156, 191075.704, 4061.906, 19205.386, 1306.137, 374.369, 372.614, 1117.539,
3014.965, 221959.299, 1001.773, 8143.973, 9892.802, 615.668, 153330.603, 659
6.874, 3495.812, 23250.762, 227.954, 4630.278, 625.37, 882.44, 771.475, 385.5
53, 1196.571, 4311.755, 45618.318, 539.071, 4661.27, 3557.317, 2785.587, 162
4.682, 6929.979, 5432.656, 23343.229, 341.469, 24923.191, 3371.362, 3593.834,
2555.525, 625.0, 1679.951, 17088.279, 1743.068, 56590.606, 1062.315, 953.944,
2025.61, 2335.775, 3502.841, 1123.858, 21260.01, 2571.228, 595.846, 4526.655,
4633.66, 507018.21, 3584.348, 1123.571, 1278.953, 6896.564, 5740.382, 8741.9
3, 10233.934]
157
```

Mapping the average sampling period with the meter id

In [138]:

```
samp_period_df = pd.DataFrame(samp_period_list, columns=['avg samp period'])
samp_period_df = samp_period_df.assign(meterid=keys_list)
samp_period_df = samp_period_df.sort_values(by=['avg samp period'], ascending=False)
samp_period_df
```

Out[138]:

	avg samp period	meterid
83	1409490.602	4874
80	560990.309	4671
149	507018.210	9620
102	221959.299	6101
94	191075.704	5545
47	169030.934	2645
107	153330.603	6685
49	96812.003	2814
137	56590.606	8703
119	45618.318	7566
52	33554.240	2946
129	24923.191	8059
127	23343.229	7965
110	23250.762	6863
48	22635.927	2755
144	21260.010	9160
72	19324.431	4228
96	19205.386	5658
50	18158.627	2818
135	17088.279	8386
71	15374.998	4193
5	15156.239	187
21	14883.727	1415
17	14797.415	1103
73	13435.344	4296
156	10233.934	9982
66	10086.732	3849
105	9892.802	6578
90	9814.518	5395
44	9514.713	2470
...	...	...
7	942.500	252
37	912.988	2094
114	882.440	7030

	avg samp period	meterid
18	856.522	1185
79	828.746	4514
115	771.475	7117
87	712.031	5193
28	646.099	1718
113	625.370	7017
133	625.000	8156
91	618.566	5403
106	615.668	6673
24	599.964	1589
146	595.846	9295
67	588.913	3893
9	572.213	483
120	539.071	7674
16	526.513	1086
12	503.056	739
22	484.959	1507
27	480.089	1714
14	450.830	871
3	435.013	94
116	385.553	7287
98	374.369	5810
99	372.614	5814
10	359.067	484
128	341.469	7989
111	227.954	6910
35	208.063	2034

157 rows × 2 columns

Getting a list containing the average sampling period across all meters as meters with the highest individual average sampling period is removed

In [139]:

```
samp_period_threshold_list = []

for i in range(156):
    sublist = samp_period_list[i:]
    samp_period_threshold_list.append(sum(sublist) / len(sublist))
```

In [140]:

```
samp_period_threshold_list.sort(reverse=True)
print(samp_period_threshold_list)
print(len(samp_period_threshold_list))
```

```
[68077.23650000001, 61027.95022222222, 55377.820700000004, 50397.64118181819,
46412.106750000006, 45049.96253246752, 44483.02385897435, 44477.330076923085,
43941.68072151898, 43452.501287499996, 43017.07745679012, 42592.38676829267,
42136.201674698794, 41794.524797619044, 41530.17075294117, 41380.65350000001,
41226.0408372093, 40766.67332183907, 40320.76304545454, 39891.797977528084, 3
9455.09925555555, 39226.99722972972, 39132.36994505494, 38855.466, 38731.7503
0666666, 38719.61338043478, 38321.578161290316, 38261.27376315788, 37932.2124
57446796, 37578.21766315789, 37260.910531249996, 36891.78814432989, 36572.985
3125, 36528.708336734686, 36173.16189898989, 35850.76249999999, 35837.7319181
8181, 35539.72059459459, 35524.63443564355, 35290.21705357142, 35221.44186274
509, 35062.1152477876, 34927.05199029126, 34765.37752631578, 34726.7004629629
5, 34621.03779807691, 34615.775935779806, 34610.87781904761, 34540.7867647058
8, 34488.29087826086, 34297.292641509426, 34238.684491379296, 34146.463990654
19, 33961.200854700845, 33731.96916101694, 33458.137638655455, 33186.92805833
3324, 32922.31834710743, 32674.851055555555, 32654.168713114745, 32406.196650
406495, 32290.012, 32152.878435483864, 31918.275335999988, 31739.80969047618,
31501.14453543306, 31264.29449999999, 31026.944147286813, 31011.033368421056,
30835.395619047624, 30791.968338461527, 30582.64711450381, 30386.14921212120
3, 30210.526681818184, 30162.192932330818, 29969.07126865671, 29750.670437037
028, 29641.35467647058, 29479.83222627736, 29275.577934782603, 29071.12429496
4022, 28970.06686956522, 28969.16922857142, 28767.44826241134, 28593.48064084
506, 28396.67888811188, 28217.59380555555, 28026.459062068956, 27864.37789041
095, 27789.02241666667, 27677.266931972783, 27494.124675675666, 27338.7296241
61064, 27162.754759999993, 27021.20541059602, 26943.146421052625, 26806.75936
6013062, 26779.682520000002, 26635.514259740252, 26473.217161290315, 26354.71
8833333325, 26195.335528662414, 25887.91911538462, 25053.97255555556, 25049.3
01785714288, 24197.307551724138, 24168.83826666667, 23564.44529032258, 23143.
835079365082, 23044.61821875, 22902.32446875, 22585.688092307693, 22500.59981
8181818, 22395.529242424243, 22252.852909090907, 22152.642053571424, 22067.20
6119402985, 21818.76623529412, 21783.605157894737, 21764.58463235294, 21549.2
53347826085, 21414.45013793103, 21297.01054285714, 21251.578742857146, 21057.
83859322034, 20966.923295774646, 20953.000736842107, 20834.90663888889, 2072
8.643566666662, 20724.66416666667, 20703.67213114754, 20526.30212820513, 2045
6.262917808217, 20435.2565483871, 20286.370540540538, 20043.05885, 19918.389
5, 19563.60748780488, 19539.904764705887, 19354.38355769231, 19142.8663773584
92, 19116.17576190476, 18806.920203703707, 18692.13539534884, 18281.527090909
09, 17978.166, 17712.68480851064, 17592.291826086956, 17416.499958333334, 171
95.691265306123, 9487.932, 8238.748666666666, 7903.202499999999, 6578.3526, 5
669.222333333334, 5371.383142857143]
156
```

In [141]:

```
samp_period_threshold_df = pd.DataFrame(samp_period_threshold_list, columns=['average samp
le rate'])
samp_period_threshold_df
```

Out[141]:

	average sample rate
0	68077.236500
1	61027.950222
2	55377.820700
3	50397.641182
4	46412.106750
5	45049.962532
6	44483.023859
7	44477.330077
8	43941.680722
9	43452.501287
10	43017.077457
11	42592.386768
12	42136.201675
13	41794.524798
14	41530.170753
15	41380.653500
16	41226.040837
17	40766.673322
18	40320.763045
19	39891.797978
20	39455.099256
21	39226.997230
22	39132.369945
23	38855.466000
24	38731.750307
25	38719.613380
26	38321.578161
27	38261.273763
28	37932.212457
29	37578.217663
...	...
126	20953.000737
127	20834.906639
128	20728.643567

	average sample rate
129	20724.664167
130	20703.672131
131	20526.302128
132	20456.262918
133	20435.256548
134	20286.370541
135	20043.058850
136	19918.389500
137	19563.607488
138	19539.904765
139	19354.383558
140	19142.866377
141	19116.175762
142	18806.920204
143	18692.135395
144	18281.527091
145	17978.166000
146	17712.684809
147	17592.291826
148	17416.499958
149	17195.691265
150	9487.932000
151	8238.748667
152	7903.202500
153	6578.352600
154	5669.222333
155	5371.383143

156 rows × 1 columns

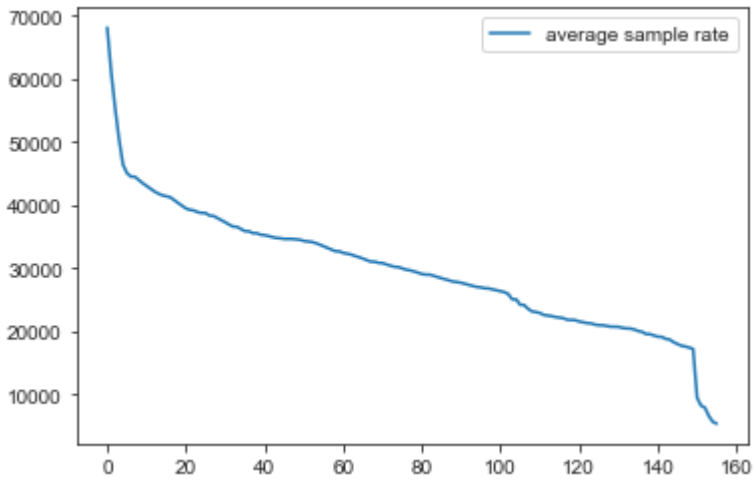
Plots of the average sampling period across all meters against number of meters (ranked based on individual average sampling period) removed

In [142]:

```
sns.lineplot(data=samp_period_threshold_df)
```

Out[142]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x22f452601d0>
```

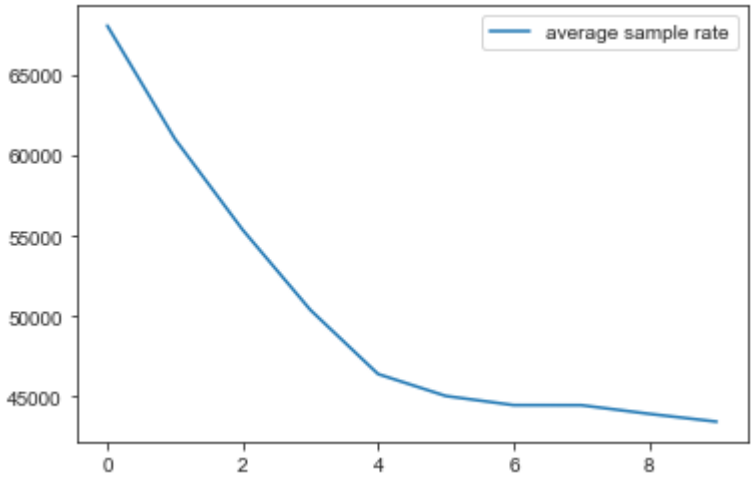


In [143]:

```
sns.lineplot(data=samp_period_threshold_df.iloc[:10])
```

Out[143]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x22f452f3c18>
```



Getting a list containing the marginal decrease in average sampling period across all meters as meters with the highest individual average sampling period is removed

In [144]:

```
samp_marginal_change_list = []

print(samp_period_threshold_list)

for i in range(len(samp_period_threshold_list)):
    if i != 0:
        samp_marginal_change_list.append(samp_period_threshold_list[i - 1] - samp_period_t
hreshold_list[i])

# print(samp_marginal_change_list)
```

```
[68077.23650000001, 61027.95022222222, 55377.820700000004, 50397.64118181819,
46412.106750000006, 45049.96253246752, 44483.02385897435, 44477.330076923085,
43941.68072151898, 43452.501287499996, 43017.07745679012, 42592.38676829267,
42136.201674698794, 41794.524797619044, 41530.17075294117, 41380.65350000001,
41226.0408372093, 40766.67332183907, 40320.76304545454, 39891.797977528084, 3
9455.09925555555, 39226.99722972972, 39132.36994505494, 38855.466, 38731.7503
0666666, 38719.61338043478, 38321.578161290316, 38261.27376315788, 37932.2124
57446796, 37578.21766315789, 37260.910531249996, 36891.78814432989, 36572.985
3125, 36528.708336734686, 36173.16189898989, 35850.76249999999, 35837.7319181
8181, 35539.72059459459, 35524.63443564355, 35290.21705357142, 35221.44186274
509, 35062.1152477876, 34927.05199029126, 34765.37752631578, 34726.7004629629
5, 34621.03779807691, 34615.775935779806, 34610.87781904761, 34540.7867647058
8, 34488.29087826086, 34297.292641509426, 34238.684491379296, 34146.463990654
19, 33961.200854700845, 33731.96916101694, 33458.137638655455, 33186.92805833
3324, 32922.31834710743, 32674.851055555555, 32654.168713114745, 32406.196650
406495, 32290.012, 32152.878435483864, 31918.275335999988, 31739.80969047618,
31501.14453543306, 31264.29449999999, 31026.944147286813, 31011.033368421056,
30835.395619047624, 30791.968338461527, 30582.64711450381, 30386.14921212120
3, 30210.526681818184, 30162.192932330818, 29969.07126865671, 29750.670437037
028, 29641.35467647058, 29479.83222627736, 29275.577934782603, 29071.12429496
4022, 28970.06686956522, 28969.16922857142, 28767.44826241134, 28593.48064084
506, 28396.67888811188, 28217.59380555555, 28026.459062068956, 27864.37789041
095, 27789.02241666667, 27677.266931972783, 27494.124675675666, 27338.7296241
61064, 27162.754759999993, 27021.20541059602, 26943.146421052625, 26806.75936
6013062, 26779.682520000002, 26635.514259740252, 26473.217161290315, 26354.71
8833333325, 26195.335528662414, 25887.91911538462, 25053.97255555556, 25049.3
01785714288, 24197.307551724138, 24168.83826666667, 23564.44529032258, 23143.
835079365082, 23044.61821875, 22902.32446875, 22585.688092307693, 22500.59981
8181818, 22395.529242424243, 22252.852909090907, 22152.642053571424, 22067.20
6119402985, 21818.76623529412, 21783.605157894737, 21764.58463235294, 21549.2
53347826085, 21414.45013793103, 21297.01054285714, 21251.578742857146, 21057.
83859322034, 20966.923295774646, 20953.000736842107, 20834.90663888889, 2072
8.643566666662, 20724.66416666667, 20703.67213114754, 20526.30212820513, 2045
6.262917808217, 20435.2565483871, 20286.370540540538, 20043.05885, 19918.389
5, 19563.60748780488, 19539.904764705887, 19354.38355769231, 19142.8663773584
92, 19116.17576190476, 18806.920203703707, 18692.13539534884, 18281.527090909
09, 17978.166, 17712.68480851064, 17592.291826086956, 17416.499958333334, 171
95.691265306123, 9487.932, 8238.748666666666, 7903.202499999999, 6578.3526, 5
669.222333333334, 5371.383142857143]
```

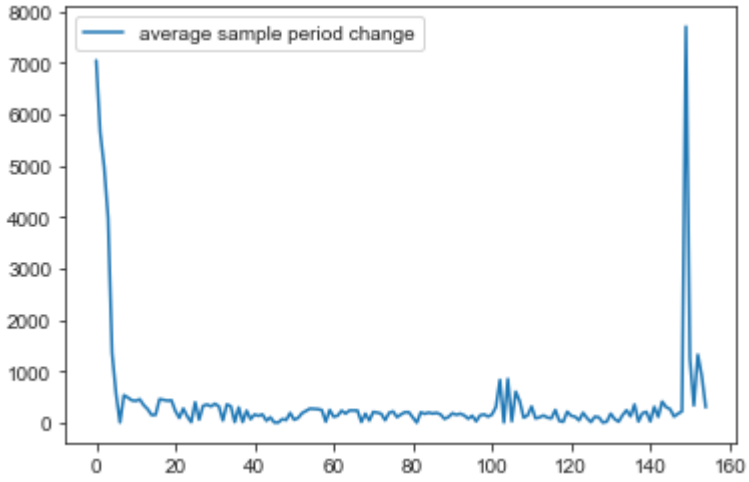
Plots of the marginal decrease in average sampling period across all meters against number of meters (ranked based on individual average sampling period) removed

In [145]:

```
samp_period_change_df = pd.DataFrame(samp_marginal_change_list, columns=['average sample p
eriod change'])
sns.lineplot(data=samp_period_change_df)
```

Out[145]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x22f45273e48>
```

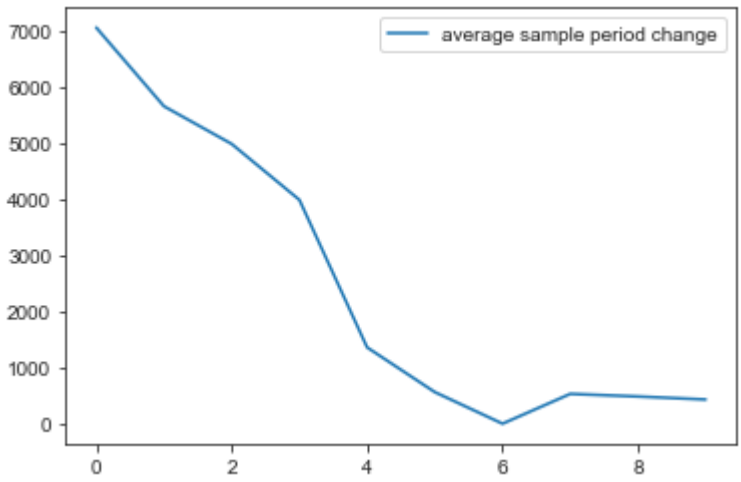


In [146]:

```
sns.lineplot(data=samp_period_change_df.iloc[:10])
```

Out[146]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x22f453c8588>
```



In [147]:

```
malfunction_samp_period = [1874, 4671, 9620, 6101, 5545, 2645]
malfunction_samp_period.sort()
print(malfunction_samp_period)
```

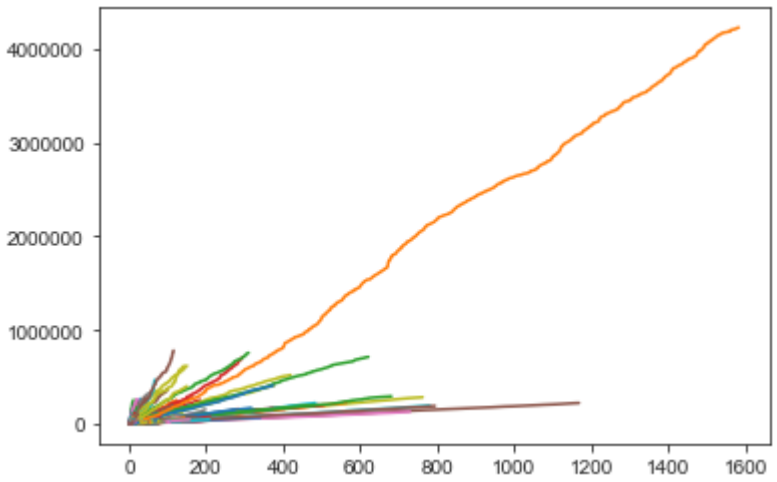
```
[1874, 2645, 4671, 5545, 6101, 9620]
```

Checking sampling of constant readings

In [208]:

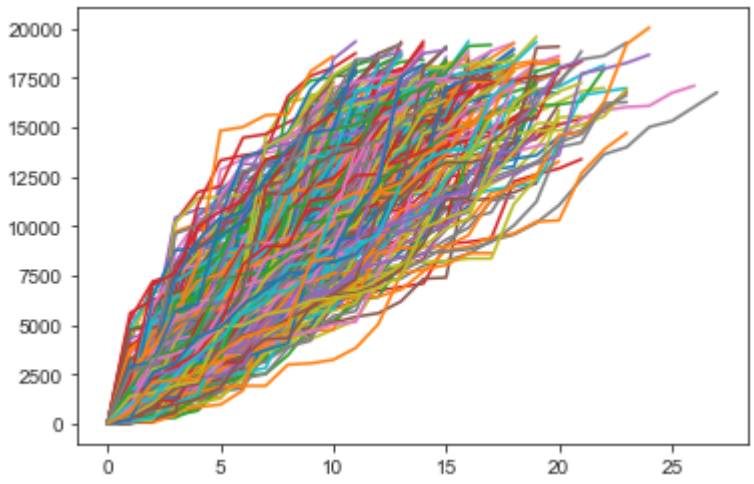
```
import matplotlib.pyplot as plt

# df_i = groups.get_group(35)
# prev_val = 0
# prev_time = df_i.index[0]
# tmp_x = []
# tmp_y = []
# constant_flag = False
# constant_count = -1

# for i in range(len(df_i.index)):
#     curr_val = df_i.iloc[i][1]
#     curr_time = df_i.index[i]
#     if i != 0:
#         if curr_val == prev_val:
#             if not constant_flag:
#                 prev_time = curr_time
#             time = curr_time - prev_time
#             time_seconds = time.total_seconds()
#             constant_count += 1
#             tmp_x.append(constant_count)
#             tmp_y.append(time_seconds)
#             constant_flag = True
#         else:
#             if len(tmp_x) > 10:
#                 plt.plot(tmp_x, tmp_y)
#             tmp_x.clear()
#             tmp_y.clear()
#             constant_flag = False
#             constant_count = -1
#     prev_val = curr_val


for key in keys_list:
    df_i = groups.get_group(key)
    prev_val = 0
    prev_time = df_i.index[0]
    tmp_x = []
    tmp_y = []
    constant_flag = False
    constant_count = -1

    for i in range(len(df_i.index)):
        curr_val = df_i.iloc[i][1]
        curr_time = df_i.index[i]
        if i != 0:
            if curr_val == prev_val:
                if not constant_flag:
                    prev_time = curr_time
                time = curr_time - prev_time
                time_seconds = time.total_seconds()
```

```
            constant_count += 1
            tmp_x.append(constant_count)
            tmp_y.append(time_seconds)
            constant_flag = True
        else:
            if len(tmp_x) > 10:
                plt.plot(tmp_x, tmp_y)
            tmp_x.clear()
            tmp_y.clear()
            constant_flag = False
            constant_count = -1
    prev_val = curr_val
```



In [209]:

```
df_i = groups.get_group(35)
prev_val = 0
prev_time = df_i.index[0]
tmp_x = []
tmp_y = []
constant_flag = False
constant_count = -1

for i in range(len(df_i.index)):
    curr_val = df_i.iloc[i][1]
    curr_time = df_i.index[i]
    if i != 0:
        if curr_val == prev_val:
            if not constant_flag:
                prev_time = curr_time
            time = curr_time - prev_time
            time_seconds = time.total_seconds()
            constant_count += 1
            tmp_x.append(constant_count)
            tmp_y.append(time_seconds)
            constant_flag = True
        else:
            if len(tmp_x) > 10:
                plt.plot(tmp_x, tmp_y)
            tmp_x.clear()
            tmp_y.clear()
            constant_flag = False
            constant_count = -1
    prev_val = curr_val
```



In [ ]: