prior_montecarlo

May 7, 2019

1 Run and process the prior monte carlo and pick a "truth" realization

A great advantage of exploring a synthetic model is that we can enforce a "truth" and then evaluate how our various attempts to estimate it perform. One way to do this is to run a monte carlo ensemble of multiple parameter realizations and then choose one of them to represent the "truth". That will be accomplished in this notebook.

In [1]:
```
import os
import shutil
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib as mpl
plt.rcParams['font.size']=12
import flopy
import pyemu
```

```
flopy is installed in /Users/jeremyw/Dev/gw1876/activities_2day_mfm/notebooks/flopy
```

1.0.1 set the `t_d` or "template directory" variable to point at the template folder and read in the PEST control file

In [2]:
```
t_d = "template"
pst = pyemu.Pst(os.path.join(t_d,"freyberg.pst"))
```

1.0.2 Decide what pars are uncertain in the truth

We need to decide what our truth looks like - should the pilot points or the grid-scale pars be the source of spatial variability? or both?

In [3]:
```
par = pst.parameter_data
# grid pars
#should_fix = par.loc[par.pargp.apply(lambda x: "gr" in x),"parnme"]
# pp pars
#should_fix = par.loc[par.pargp.apply(lambda x: "pp" in x),"parnme"]
#pst.npar - should_fix.shape[0]
```

```
In [4]: pe = pyemu.ParameterEnsemble.from_binary(pst=pst,filename=os.path.join(t_d,"prior.jcb"
        #pe.loc[:,should_fix] = 1.0
        pe.to_csv(os.path.join(t_d,"sweep_in.csv"))
```

```
new binary format detected...
```

1.0.3 run the prior ensemble in parallel locally

This takes advantage of the program `pestpp-swp` which runs a parameter sweep through a set of parameters. By default, `pestpp-swp` reads in the ensemble from a file called `sweep_in.csv` which in this case we made just above.

```
In [5]: m_d = "master_prior_sweep"
        pyemu.os_utils.start_slaves(t_d,"pestpp-swp","freyberg.pst",num_slaves=20,slave_root="
```

1.0.4 Load the output ensemble and plot a few things

```
In [6]: obs_df = pd.read_csv(os.path.join(m_d,"sweep_out.csv"),index_col=0)
        print('number of realization in the ensemble before dropping: ' + str(obs_df.shape[0])
```

```
number of realization in the ensemble before dropping: 200
```

drop any failed runs

```
In [7]: obs_df = obs_df.loc[obs_df.failed_flag==0,:]
        print('number of realization in the ensemble **after** dropping: ' + str(obs_df.shape[
```

```
number of realization in the ensemble **after** dropping: 200
```

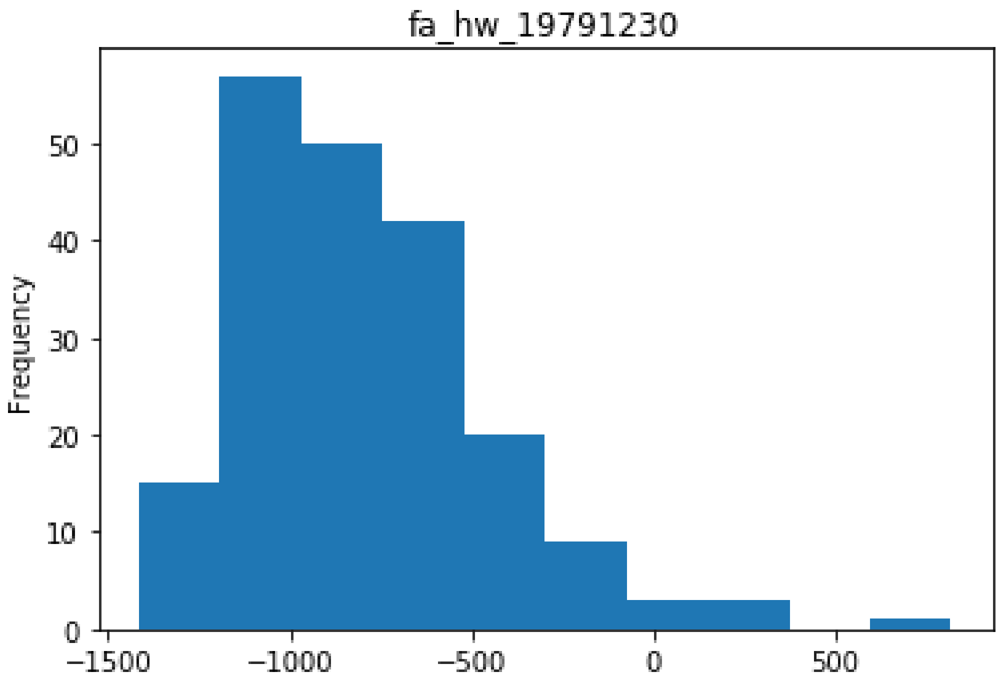
1.0.5 confirm which quantities were identified as forecasts

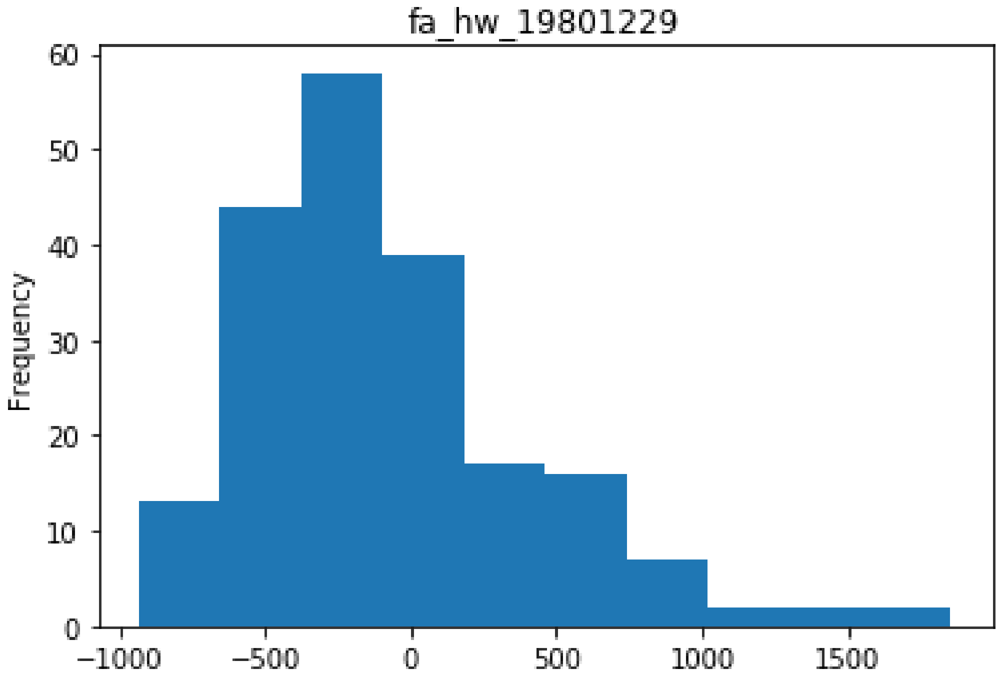
```
In [8]: fnames = pst.pestpp_options["forecasts"].split(',')
        fnames
```

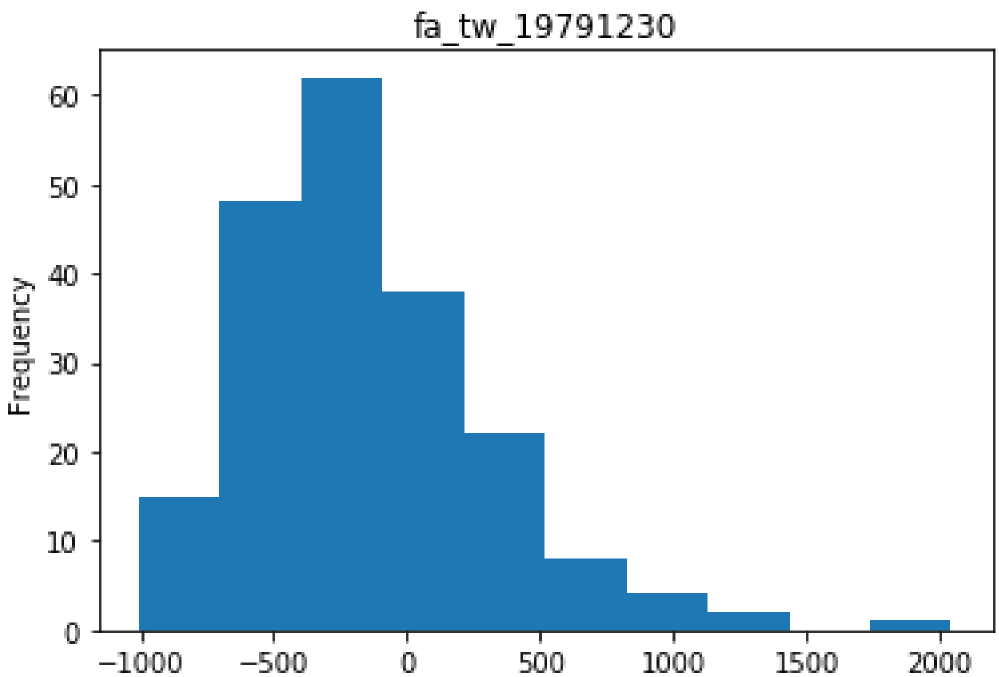
```
Out[8]: ['fa_hw_19791230',
         'fa_hw_19801229',
         'fa_tw_19791230',
         'fa_tw_19801229',
         'hds_00_013_002_000',
         'hds_00_013_002_001',
         'part_time',
         'part_status']
```

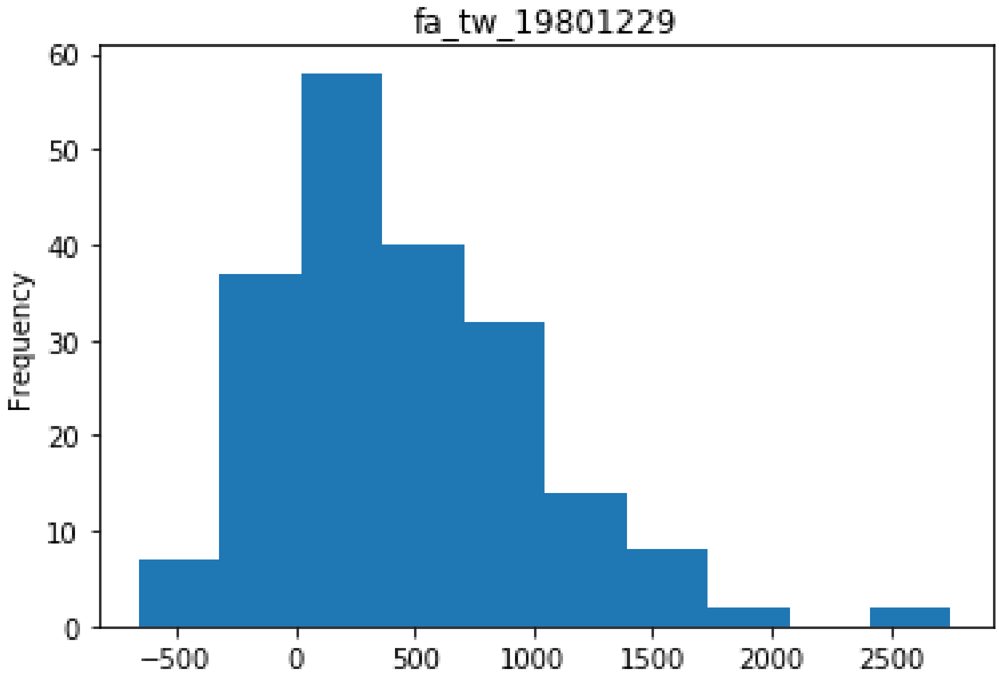
1.0.6 now we can plot the distributions of each forecast

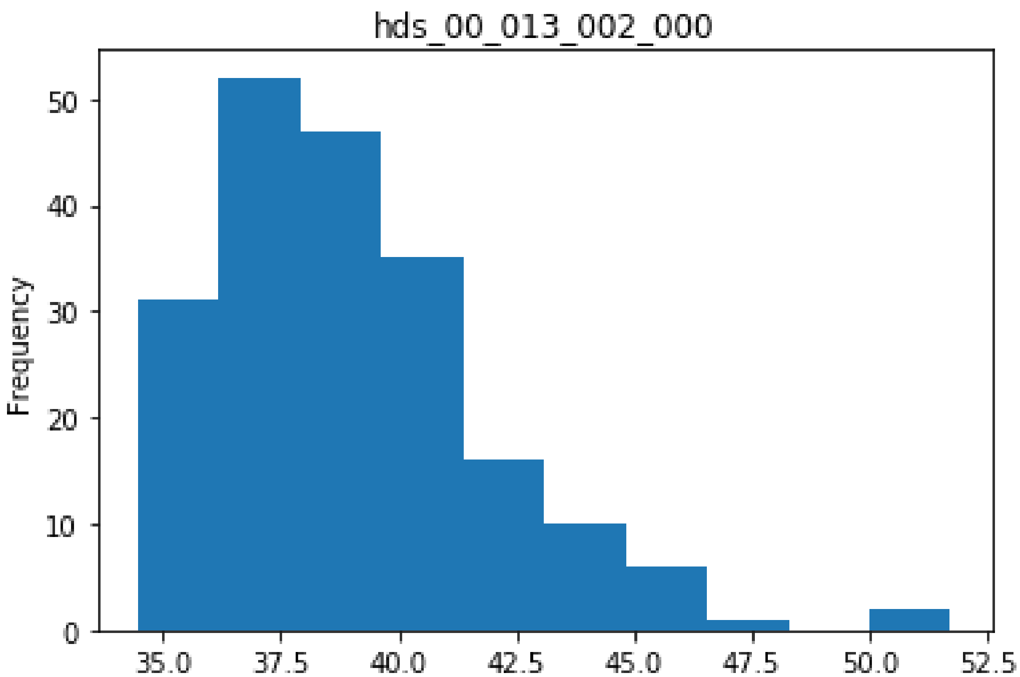
```
In [9]: for forecast in fnames:
            plt.figure()
            ax = obs_df.loc[:,forecast].plot(kind="hist")
            ax.set_title(forecast)
        plt.show()
```

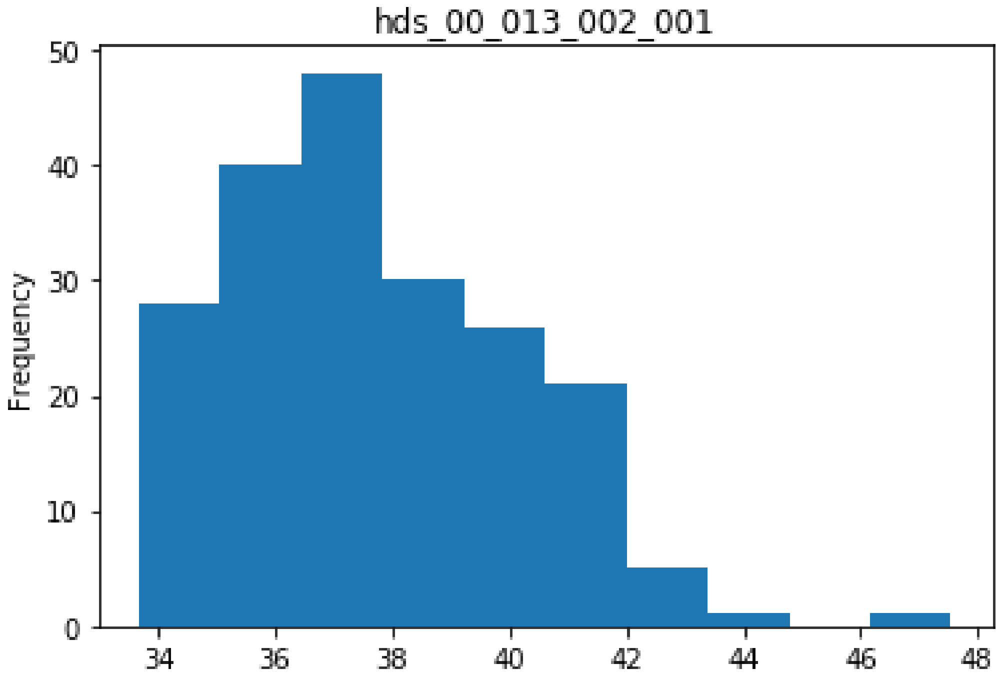


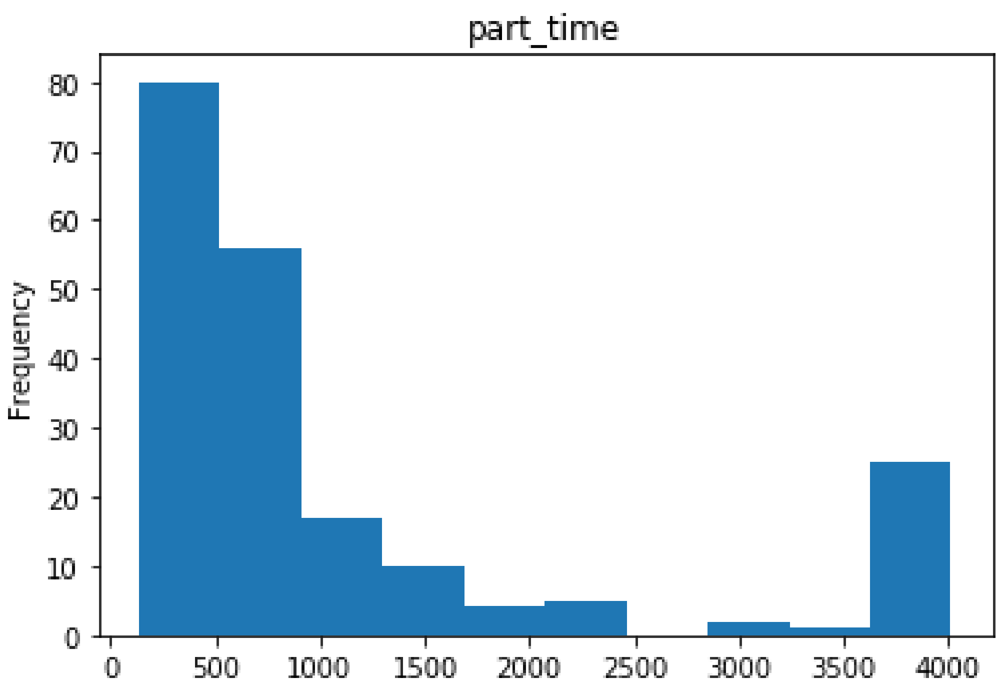

fa_hw_19801229
Frequency
60
50
40
30
20
10
0
−1000
−500
0
500
1000
1500

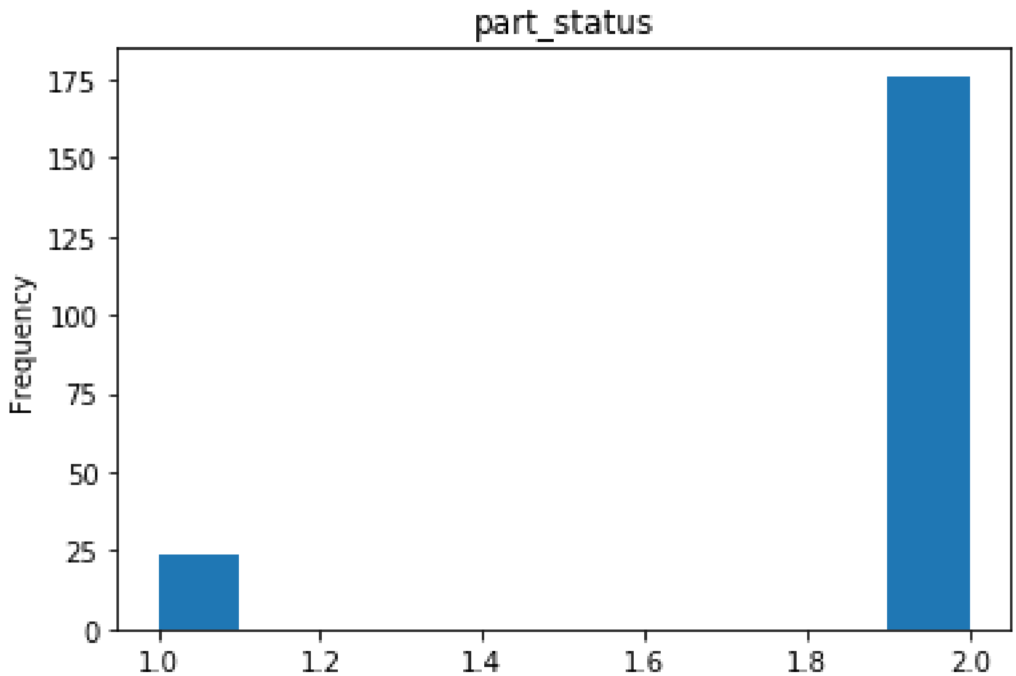

fa_tw_19791230
Frequency
60
50
40
30
20
10
0
−1000
−500
0
500
1000
1500
2000


fa_tw_19801229
Frequency
60
50
40
30
20
10
0
−500
0
500
1000
1500
2000
2500


hds_00_013_002_000
Frequency
50
40
30
20
10
0
35.0
37.5
40.0
42.5
45.0
47.5
50.0
52.5

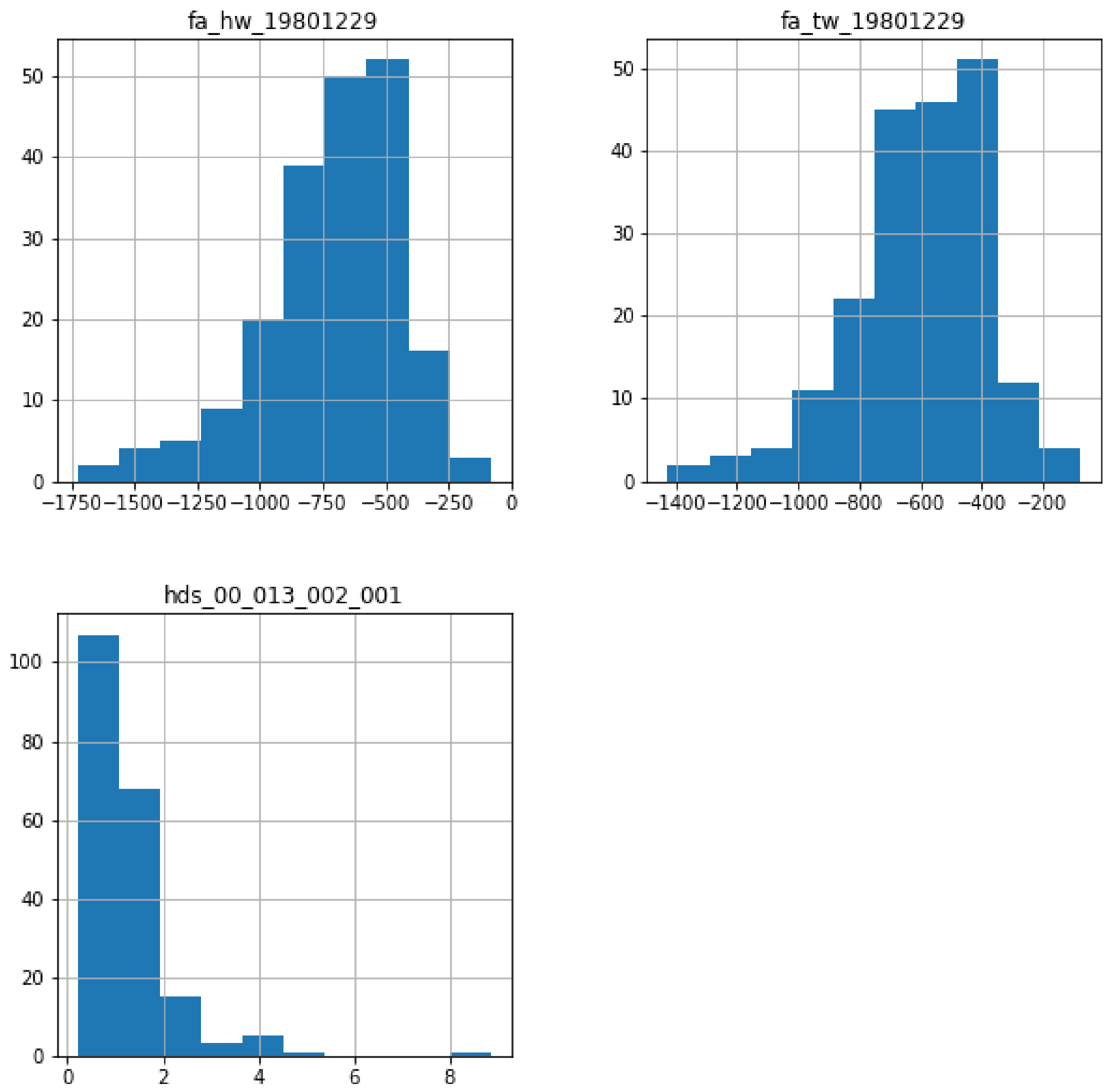

hds_00_013_002_001
Frequency
0
10
20
30
40
50
34
36
38
40
42
44
46
48


part_time
Frequency
0
10
20
30
40
50
60
70
80
0
500
1000
1500
2000
2500
3000
3500
4000



We see that under scenario conditions, many more realizations for the flow to the aquifer in the headwaters are postive (as expected). Lets difference these two:

```
In [10]: sfnames = [f for f in fnames if "1980" in f or "_001" in f]
         hfnames = [f for f in fnames if "1979" in f or "_000" in f]
         diff = obs_df.loc[:,hfnames].values - obs_df.loc[:,sfnames].values
         diff = pd.DataFrame(diff,columns=sfnames)
         diff.hist(figsize=(10,10))
         plt.show()
```



We now see that the most extreme scenario yields a large decrease in flow from the aquifer to the headwaters (the most negative value)

1.0.7 setting the "truth"

We just need to replace the observed values (`obsval`) in the control file with the outputs for one of the realizations on `obs_df`. In this way, we now have the nonzero values for history matching, but also the `truth` values for comparing how we are doing with other unobserved quantities. I'm going to pick a realization that yields an "average" variability of the observed gw levels:

```
In [11]: # choose the realization with a low historic gw to sw headwater flux
         hist_swgw = obs_df.loc[:,"fa_hw_19791230"].sort_values()
         idx = hist_swgw.index[10]
         idx
```

```
Out[11]: 193
```

```
In [12]: obs_df.loc[idx,pst.nnz_obs_names]
```

```
Out[12]: fo_39_19791230         12065.000000
         hds_00_002_009_000        40.873676
         hds_00_002_015_000        35.360447
         hds_00_003_008_000        41.228016
         hds_00_009_001_000        44.455975
         hds_00_013_010_000        37.670860
         hds_00_015_016_000        35.308788
         hds_00_021_010_000        36.780243
         hds_00_022_015_000        34.332844
         hds_00_024_004_000        39.176395
         hds_00_026_006_000        38.244102
         hds_00_029_015_000        34.879692
         hds_00_033_007_000        37.037766
         hds_00_034_010_000        36.083714
         Name: 193, dtype: float64
```

Lets see how our selected truth does with the sw/gw forecasts:

```
In [13]: obs_df.loc[idx,fnames]
```

```
Out[13]: fa_hw_19791230        -1218.824400
         fa_hw_19801229         -890.556500
         fa_tw_19791230        -1005.060000
         fa_tw_19801229         -622.118360
         hds_00_013_002_000       44.474049
         hds_00_013_002_001       42.691513
         part_time               948.015500
         part_status               2.000000
         Name: 193, dtype: float64
```

Assign some initial weights. Now, it is custom to add noise to the observed values…we will use the classic Gaussian noise…zero mean and standard deviation of 1 over the weight

```
In [14]: pst = pyemu.Pst(os.path.join(t_d,"freyberg.pst"))
         obs = pst.observation_data
         obs.loc[:,"obsval"] = obs_df.loc[idx,pst.obs_names]
         obs.loc[obs.obgnme=="calhead","weight"] = 10.0
         obs.loc[obs.obgnme=="calflux","weight"] = 0.05
```

here we just get a sample from a random normal distribution with mean=0 and std=1. The argument indicates how many samples we want - and we choose `pst.nnz_obs` which is the the number of nonzero-weighted observations in the PST file

```
In [15]: np.random.seed(seed=0)
         snd = np.random.randn(pst.nnz_obs)
         noise = snd * 1./obs.loc[pst.nnz_obs_names,"weight"]
         pst.observation_data.loc[noise.index,"obsval"] += noise
         noise
```

```
Out[15]: obsnme
         fo_39_19791230        35.281047
         hds_00_002_009_000     0.040016
         hds_00_002_015_000     0.097874
         hds_00_003_008_000     0.224089
         hds_00_009_001_000     0.186756
         hds_00_013_010_000    -0.097728
         hds_00_015_016_000     0.095009
         hds_00_021_010_000    -0.015136
         hds_00_022_015_000    -0.010322
         hds_00_024_004_000     0.041060
         hds_00_026_006_000     0.014404
         hds_00_029_015_000     0.145427
         hds_00_033_007_000     0.076104
         hds_00_034_010_000     0.012168
         Name: weight, dtype: float64
```

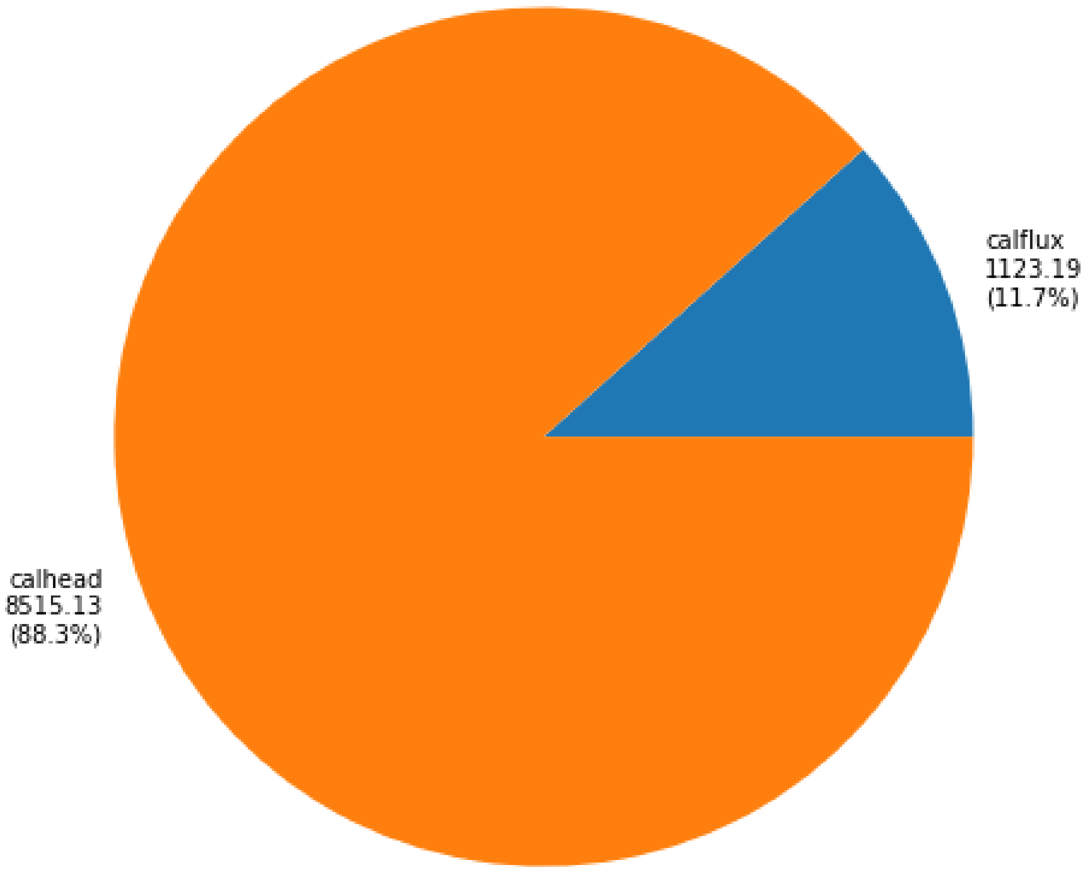
Then we write this out to a new file and run `pestpp-ies` to see how the objective function looks

```
In [16]: pst.write(os.path.join(t_d,"freyberg.pst"))
         pyemu.os_utils.run("pestpp-ies freyberg.pst",cwd=t_d)
```

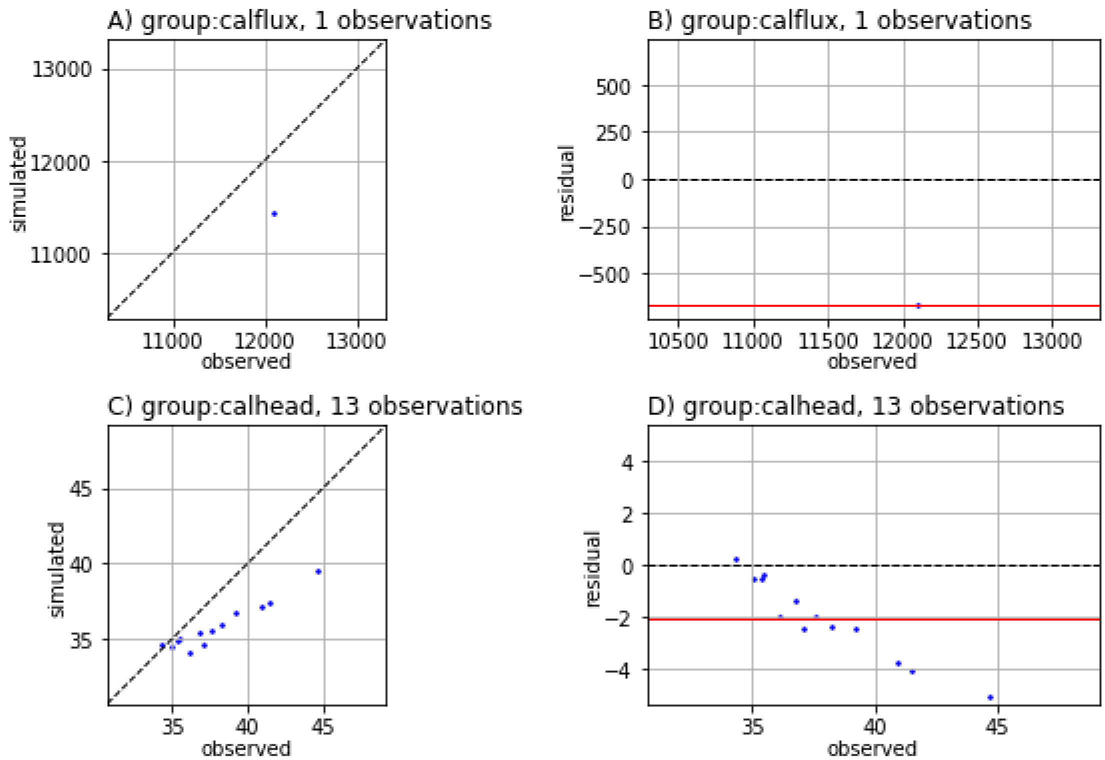
Now we can read in the results and make some figures showing residuals and the balance of the objective function

```
In [17]: pst = pyemu.Pst(os.path.join(t_d,"freyberg.pst"))
         print(pst.phi)
         plt.figure()
         pst.plot(kind='phi_pie')
         print('Here are the non-zero weighted observation names')

         figs = pst.plot(kind="1to1")
         plt.show()
         pst.res.loc[pst.nnz_obs_names,:]
```

```
9638.320232180446
Here are the non-zero weighted observation names


<Figure size 432x288 with 0 Axes>
```




```
<Figure size 576x756 with 0 Axes>
```




```
Out[17]:                                        name     group      measured      modelled  \
         name
         fo_39_19791230           fo_39_19791230   calflux  12100.281047  11430.000000
```

```
hds_00_002_009_000  hds_00_002_009_000  calhead     40.913692     37.107498
hds_00_002_015_000  hds_00_002_015_000  calhead     35.458321     35.045185
hds_00_003_008_000  hds_00_003_008_000  calhead     41.452105     37.397289
hds_00_009_001_000  hds_00_009_001_000  calhead     44.642730     39.546417
hds_00_013_010_000  hds_00_013_010_000  calhead     37.573133     35.571774
hds_00_015_016_000  hds_00_015_016_000  calhead     35.403797     34.835716
hds_00_021_010_000  hds_00_021_010_000  calhead     36.765107     35.386250
hds_00_022_015_000  hds_00_022_015_000  calhead     34.322522     34.577492
hds_00_024_004_000  hds_00_024_004_000  calhead     39.217455     36.760464
hds_00_026_006_000  hds_00_026_006_000  calhead     38.258507     35.896149
hds_00_029_015_000  hds_00_029_015_000  calhead     35.025119     34.453842
hds_00_033_007_000  hds_00_033_007_000  calhead     37.113869     34.678810
hds_00_034_010_000  hds_00_034_010_000  calhead     36.095881     34.118073

                      residual  weight
name
fo_39_19791230      670.281047    0.05
hds_00_002_009_000    3.806194   10.00
hds_00_002_015_000    0.413136   10.00
hds_00_003_008_000    4.054816   10.00
hds_00_009_001_000    5.096313   10.00
hds_00_013_010_000    2.001359   10.00
hds_00_015_016_000    0.568081   10.00
hds_00_021_010_000    1.378858   10.00
hds_00_022_015_000   -0.254970   10.00
hds_00_024_004_000    2.456992   10.00
hds_00_026_006_000    2.362358   10.00
hds_00_029_015_000    0.571277   10.00
hds_00_033_007_000    2.435059   10.00
hds_00_034_010_000    1.977809   10.00
```

Publication ready figs - oh snap!

Depending on the truth you chose, we may have a problem - we set the weights for both the heads and the flux to reasonable values based on what we expect for measurement noise. But the contributions to total phi might be out of balance - if contribution of the flux measurement to total phi is too low, the history matching excersizes (coming soon!) will focus almost entirely on minimizing head residuals. So we need to balance the objective function. This is a subtle but very important step, especially since some of our forecasts deal with sw-gw exchange

In [18]:
```
#pc = pst.phi_components
#target = {"calflux":0.3 * pc["calhead"]}
#pst.adjust_weights(obsgrp_dict=target)
#pst.plot(kind='phi_pie')
```

Just to make sure we have everything working right, we should be able to load the truth parameters, run the model once and have a `phi` equivalent to the noise vector:

In [19]:
```
par_df = pd.read_csv(os.path.join(m_d,"sweep_in.csv"),index_col=0)
pst.parameter_data.loc[:,"parval1"] = par_df.loc[idx,pst.par_names]
pst.write(os.path.join(m_d,"test.pst"))
```

we will run this with `noptmax=0` to preform a single run. Pro-tip: you can use any of the `pestpp-###` binaries/executables to run `noptmax=0`

```
In [20]: pyemu.os_utils.run("pestpp-ies.exe test.pst",cwd=m_d)
         pst = pyemu.Pst(os.path.join(m_d,"test.pst"))
         print(pst.phi)
         pst.res.loc[pst.nnz_obs_names,:]
```

```
17.528847219729652
```

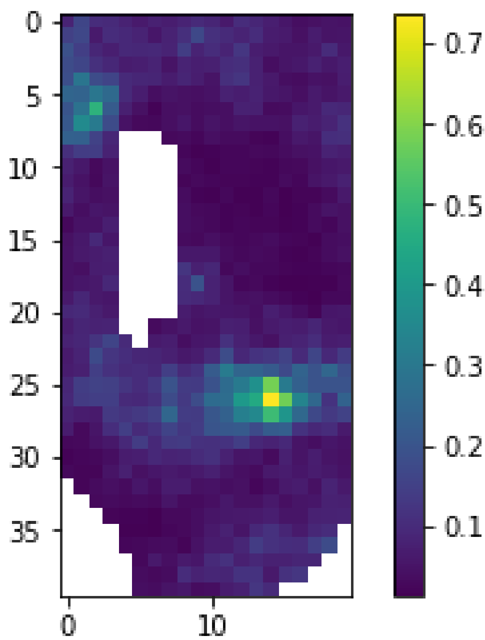
```
Out[20]:                                   name    group      measured      modelled  \
         name
         fo_39_19791230          fo_39_19791230  calflux  12100.281047  12065.000000
         hds_00_002_009_000  hds_00_002_009_000  calhead     40.913692     40.873676
         hds_00_002_015_000  hds_00_002_015_000  calhead     35.458321     35.360447
         hds_00_003_008_000  hds_00_003_008_000  calhead     41.452105     41.228016
         hds_00_009_001_000  hds_00_009_001_000  calhead     44.642730     44.455975
         hds_00_013_010_000  hds_00_013_010_000  calhead     37.573133     37.670860
         hds_00_015_016_000  hds_00_015_016_000  calhead     35.403797     35.308788
         hds_00_021_010_000  hds_00_021_010_000  calhead     36.765107     36.780243
         hds_00_022_015_000  hds_00_022_015_000  calhead     34.322522     34.332844
         hds_00_024_004_000  hds_00_024_004_000  calhead     39.217455     39.176395
         hds_00_026_006_000  hds_00_026_006_000  calhead     38.258507     38.244102
         hds_00_029_015_000  hds_00_029_015_000  calhead     35.025119     34.879692
         hds_00_033_007_000  hds_00_033_007_000  calhead     37.113869     37.037766
         hds_00_034_010_000  hds_00_034_010_000  calhead     36.095881     36.083714

                              residual  weight
         name
         fo_39_19791230      35.281047    0.05
         hds_00_002_009_000   0.040016   10.00
         hds_00_002_015_000   0.097874   10.00
         hds_00_003_008_000   0.224089   10.00
         hds_00_009_001_000   0.186756   10.00
         hds_00_013_010_000  -0.097728   10.00
         hds_00_015_016_000   0.095009   10.00
         hds_00_021_010_000  -0.015136   10.00
         hds_00_022_015_000  -0.010322   10.00
         hds_00_024_004_000   0.041060   10.00
         hds_00_026_006_000   0.014404   10.00
         hds_00_029_015_000   0.145427   10.00
         hds_00_033_007_000   0.076104   10.00
         hds_00_034_010_000   0.012168   10.00
```

The residual should be exactly the noise values from above.Lets load the model (that was just run using the true pars) and check some things

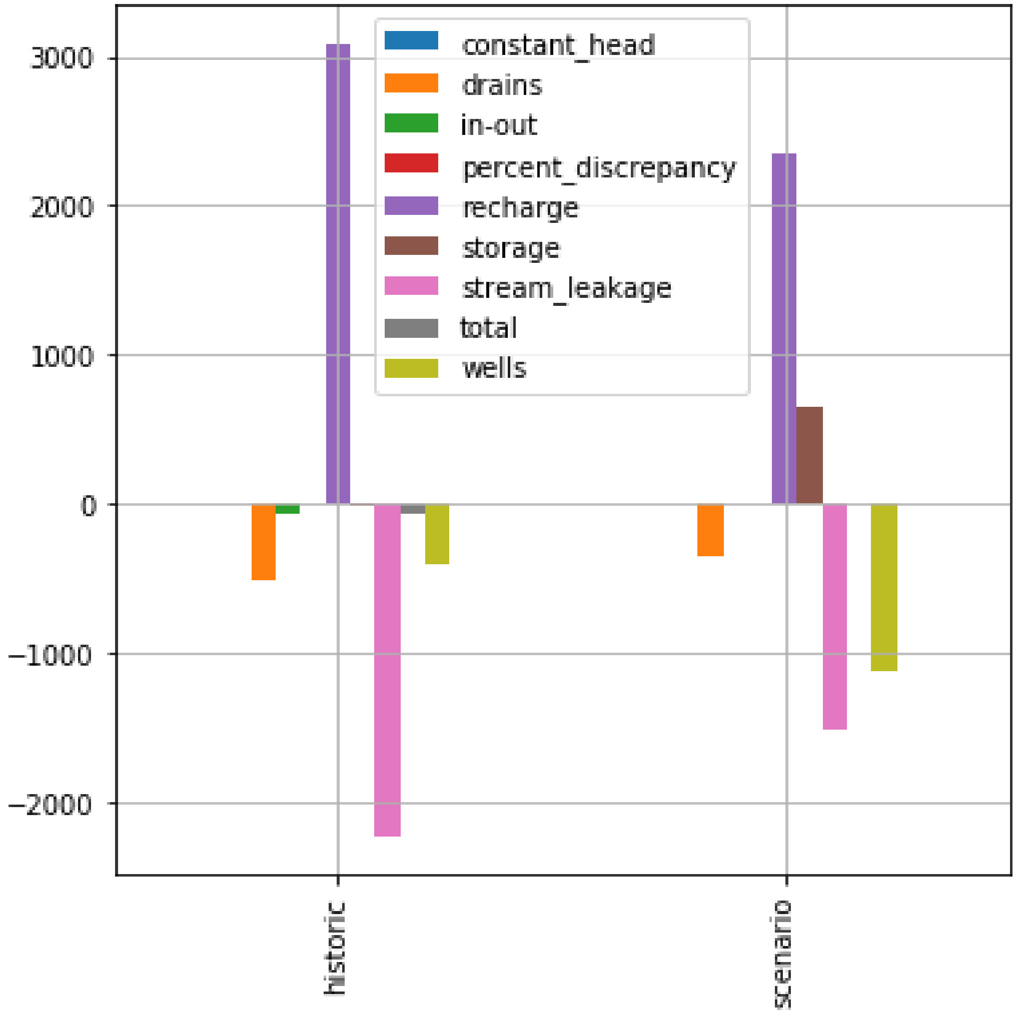
```
In [21]: m = flopy.modflow.Modflow.load("freyberg.nam",model_ws=m_d)
```

```
In [22]: a = m.upw.vka[1].array
         #a = m.rch.rech[0].array
         a = np.ma.masked_where(m.bas6.ibound[0].array==0,a)
         print(a.min(),a.max())
         c = plt.imshow(a)
         plt.colorbar()
         plt.show()
```

```
0.01317227 0.735812
```



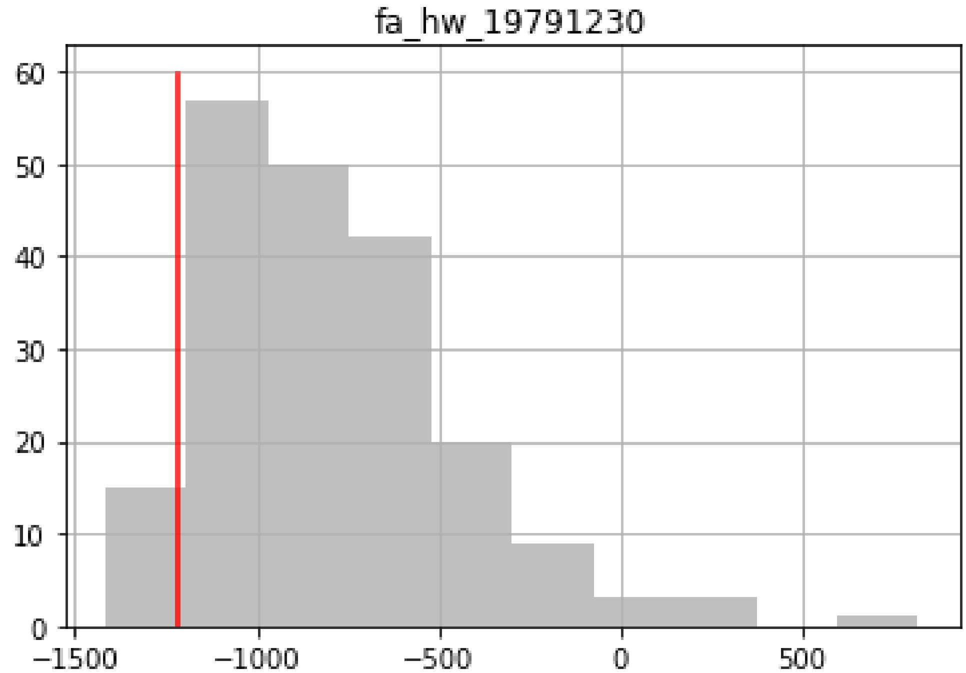

```
In [23]: lst = flopy.utils.MfListBudget(os.path.join(m_d,"freyberg.list"))
         df = lst.get_dataframes(diff=True)[0]
         ax = df.plot(kind="bar",figsize=(6,6), grid=True)
         a = ax.set_xticklabels(["historic","scenario"],rotation=90)
         plt.show(ax)
```

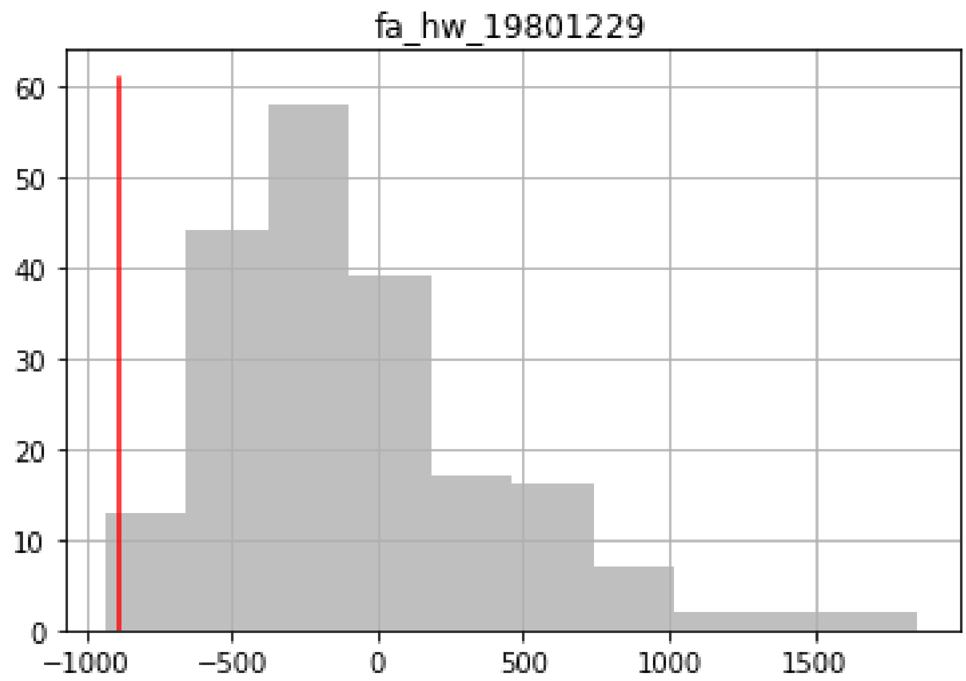


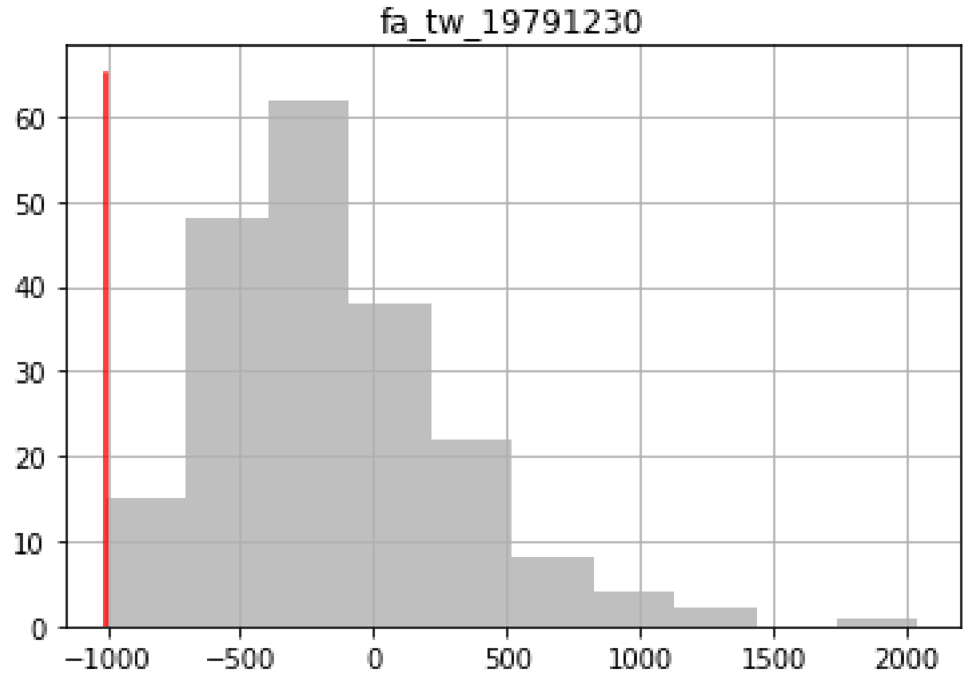
1.0.8 see how our existing observation ensemble compares to the truth

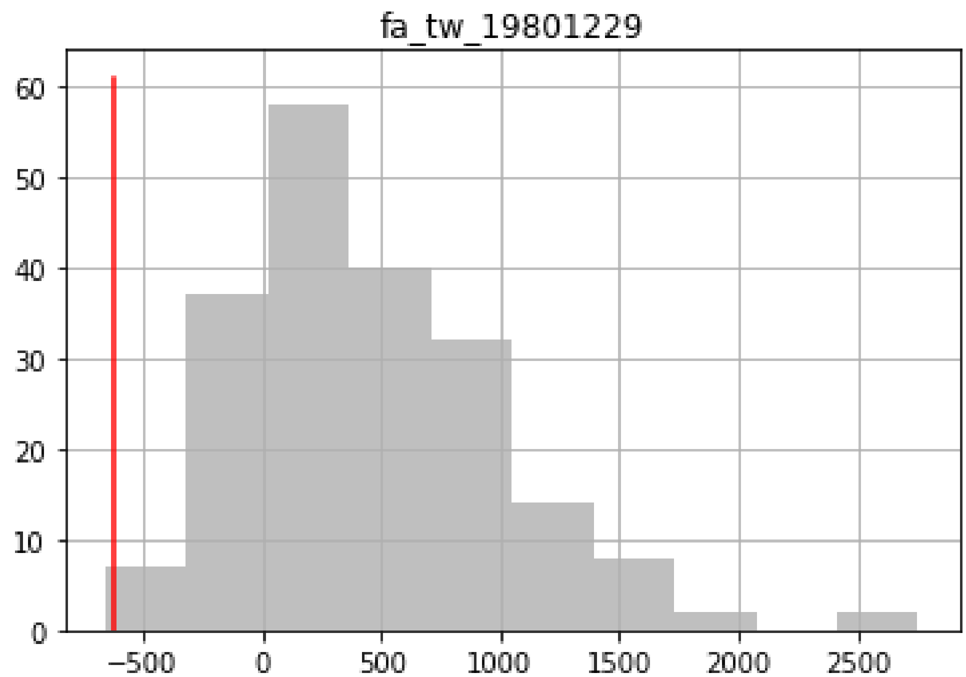
forecasts:

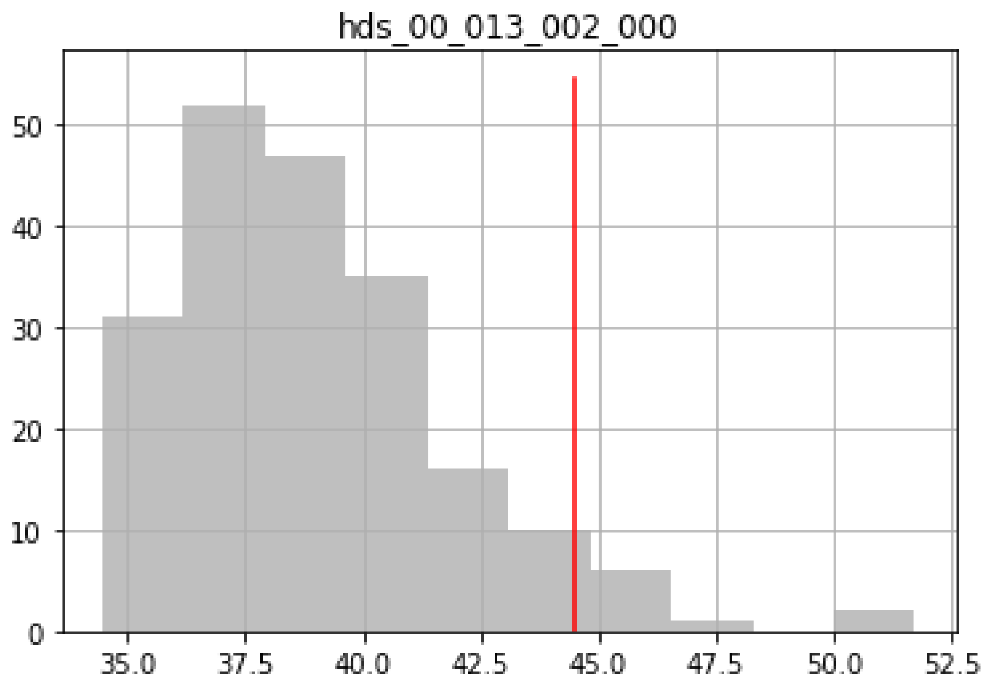
```
In [24]: obs = pst.observation_data
         plt.figure()
         for forecast in fnames:
             ax = plt.subplot(111)
             obs_df.loc[:,forecast].hist(ax=ax,color="0.5",alpha=0.5)
             ax.plot([obs.loc[forecast,"obsval"],obs.loc[forecast,"obsval"]],ax.get_ylim(),"r"
             ax.set_title(forecast)
             plt.show()
```

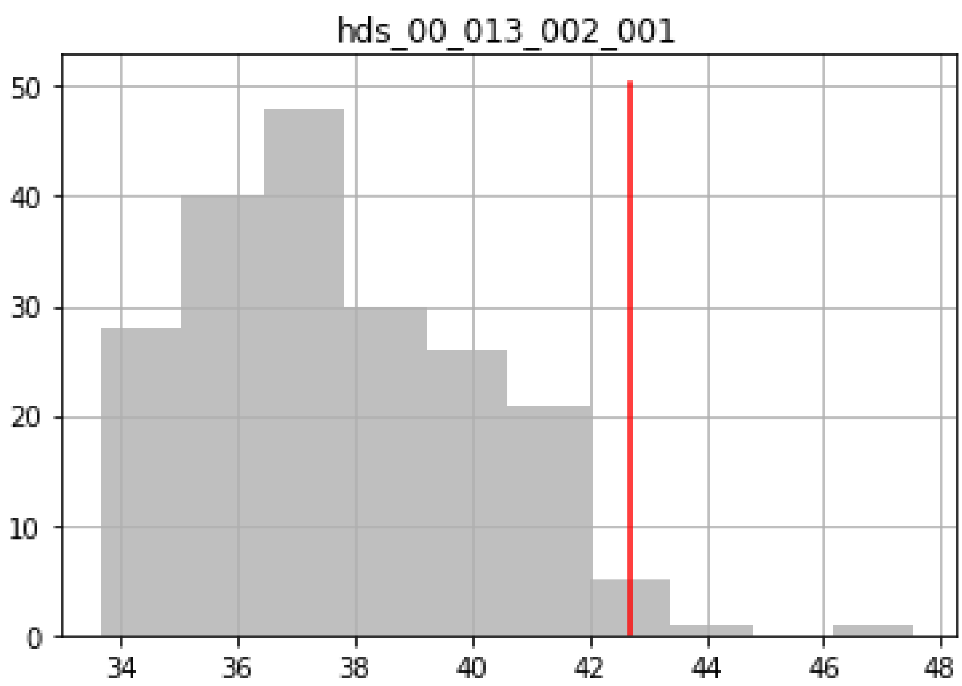

fa_hw_19791230
60
50
40
30
20
10
0
−1500
−1000
−500
0
500

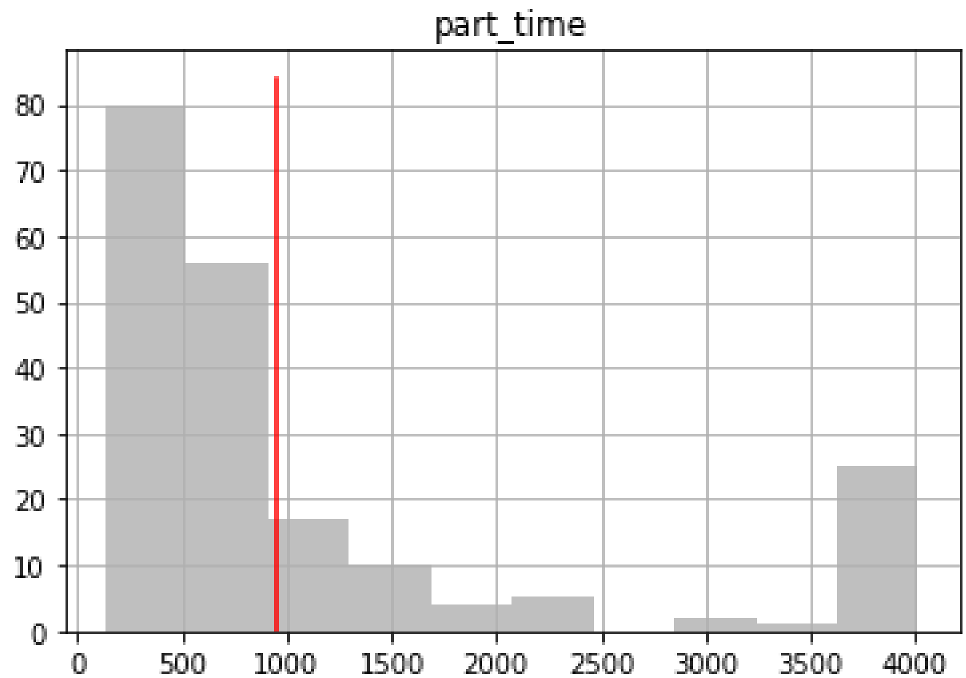

fa_hw_19801229
60
50
40
30
20
10
0
−1000
−500
0
500
1000
1500

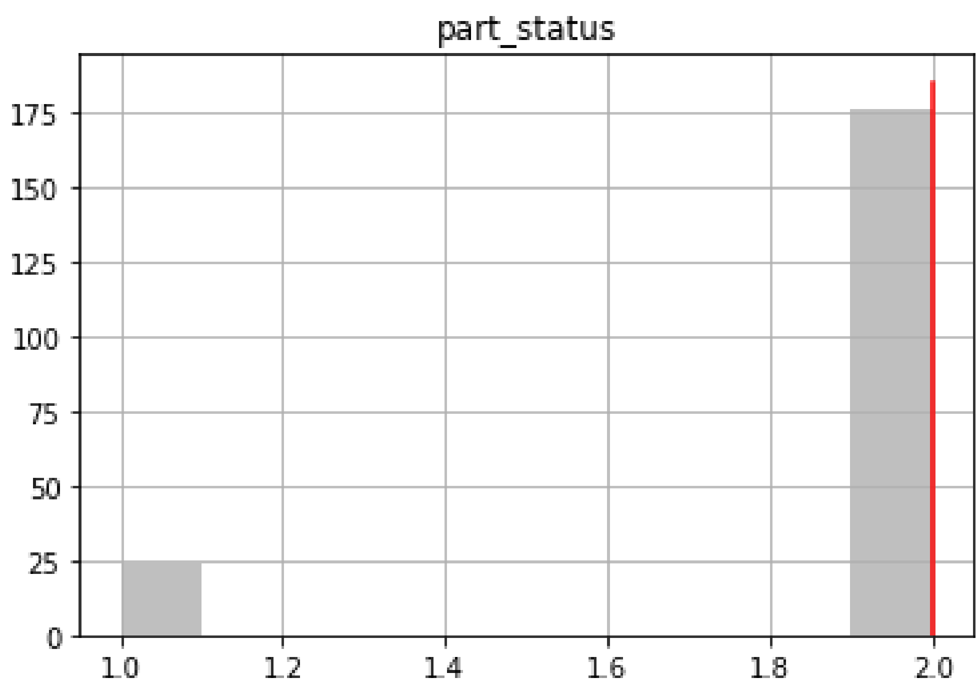

fa_tw_19791230
60
50
40
30
20
10
0
−1000
−500
0
500
1000
1500
2000


fa_tw_19801229
60
50
40
30
20
10
0
−500
0
500
1000
1500
2000
2500


hds_00_013_002_000
0
10
20
30
40
50
35.0
37.5
40.0
42.5
45.0
47.5
50.0
52.5

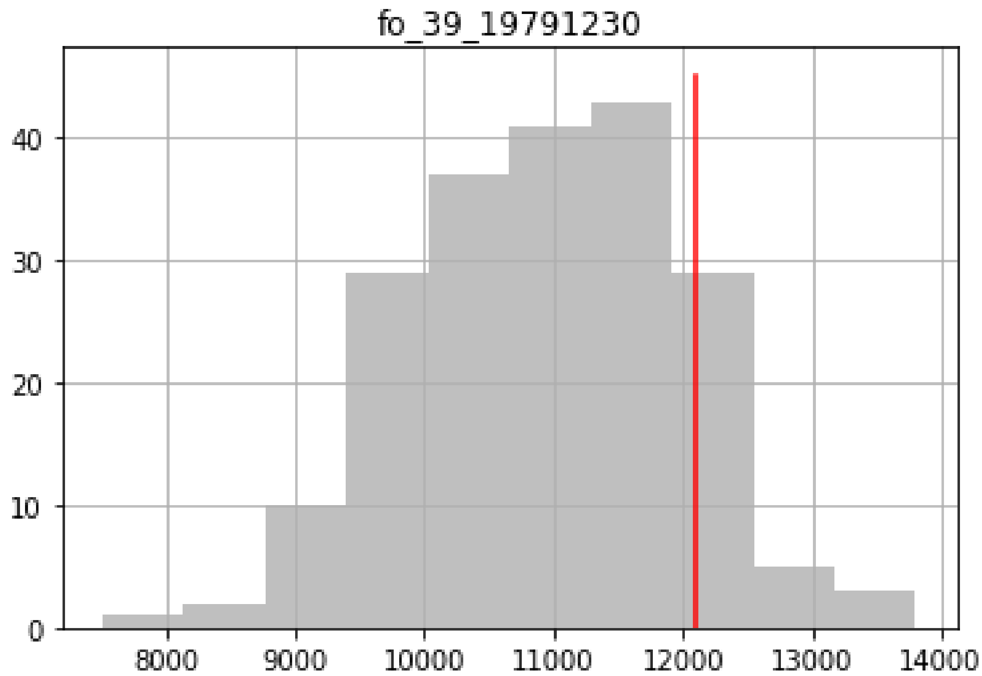

hds_00_013_002_001
0
10
20
30
40
50
34
36
38
40
42
44
46
48

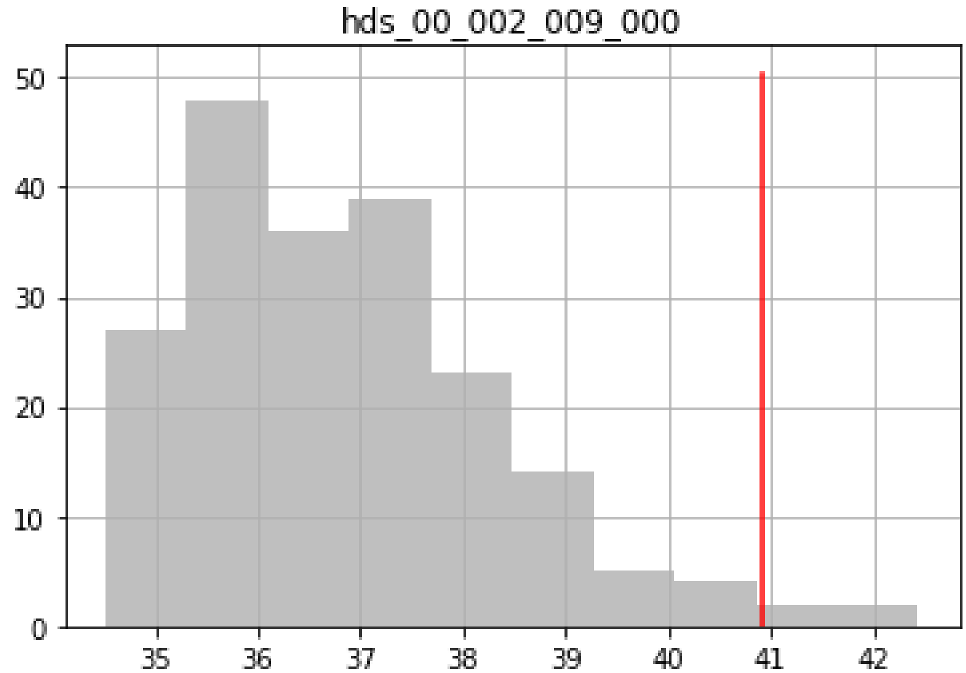


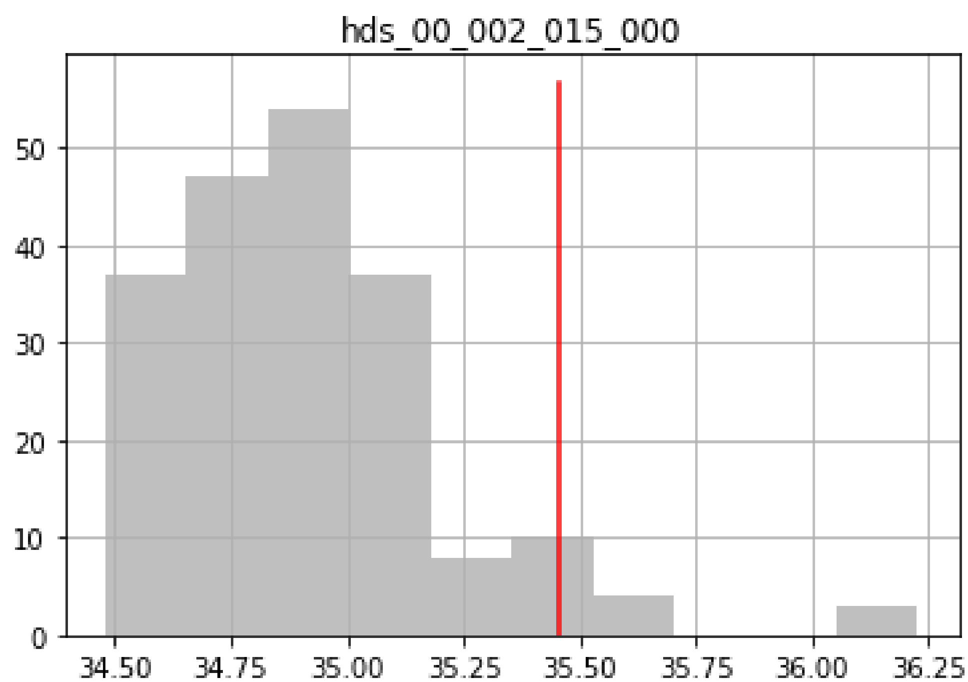


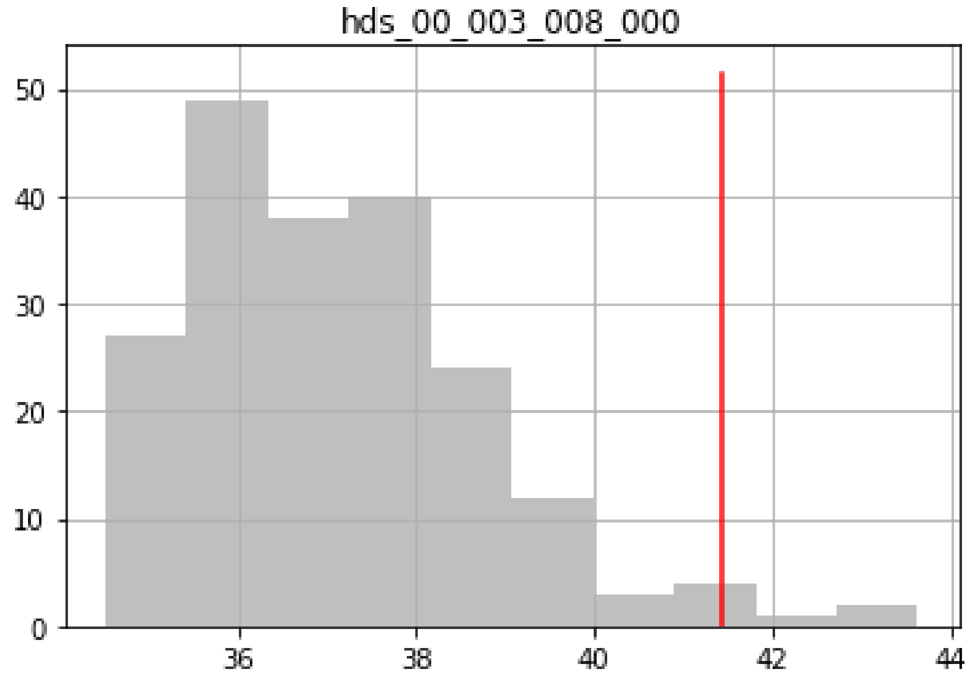
observations:

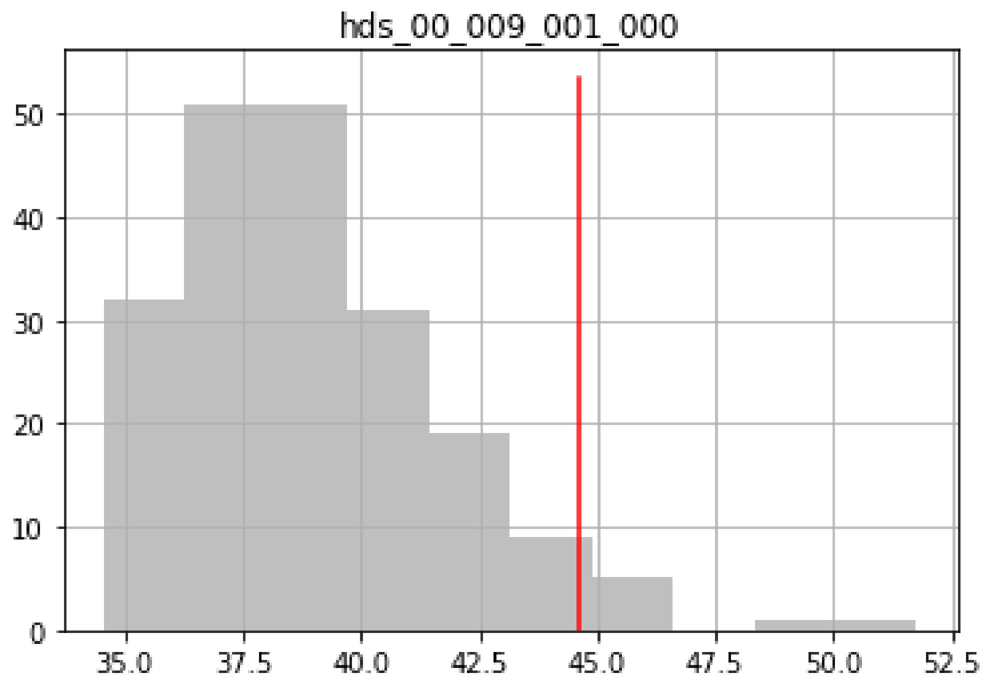
In [25]:
```
for oname in pst.nnz_obs_names:
    ax = plt.subplot(111)
    obs_df.loc[:,oname].hist(ax=ax,color="0.5",alpha=0.5)
    ax.plot([obs.loc[oname,"obsval"],obs.loc[oname,"obsval"]],ax.get_ylim(),"r")
    ax.set_title(oname)
    plt.show()
```

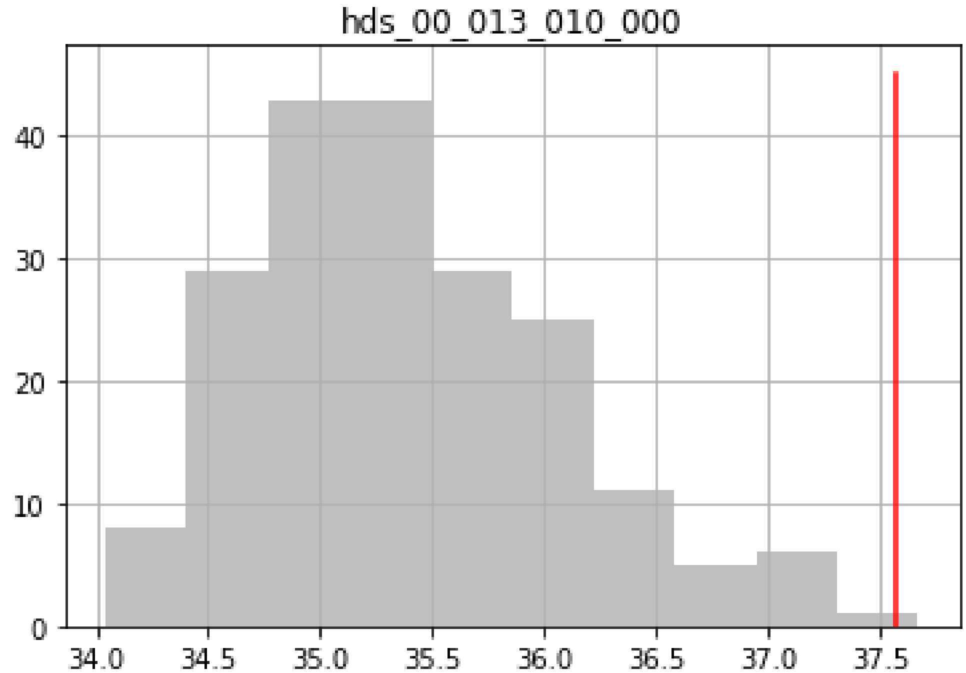


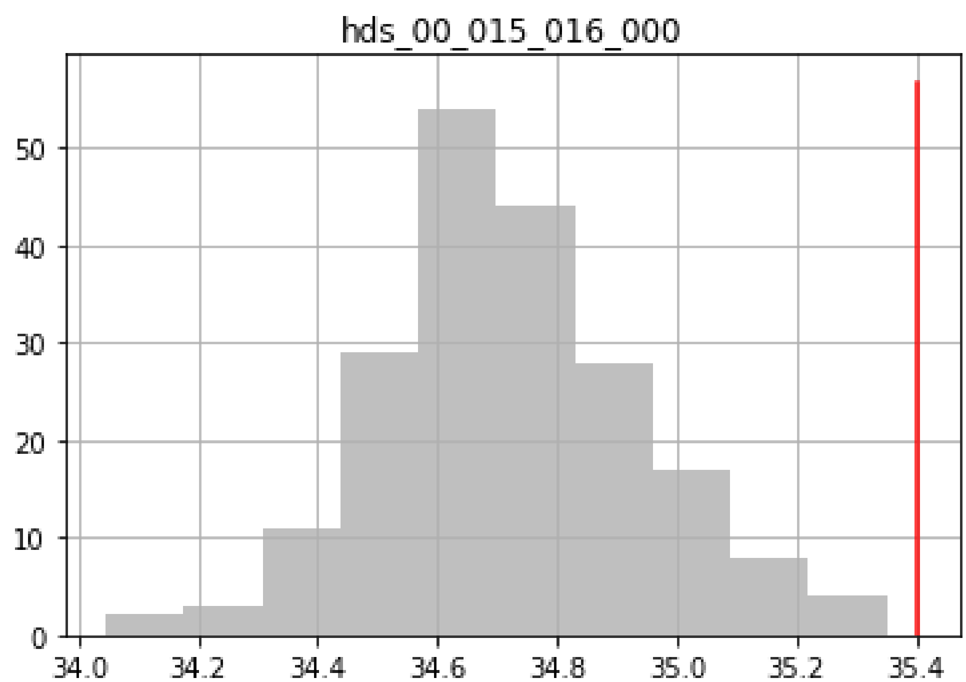

hds_00_002_009_000
50
40
30
20
10
0
35
36
37
38
39
40
41
42

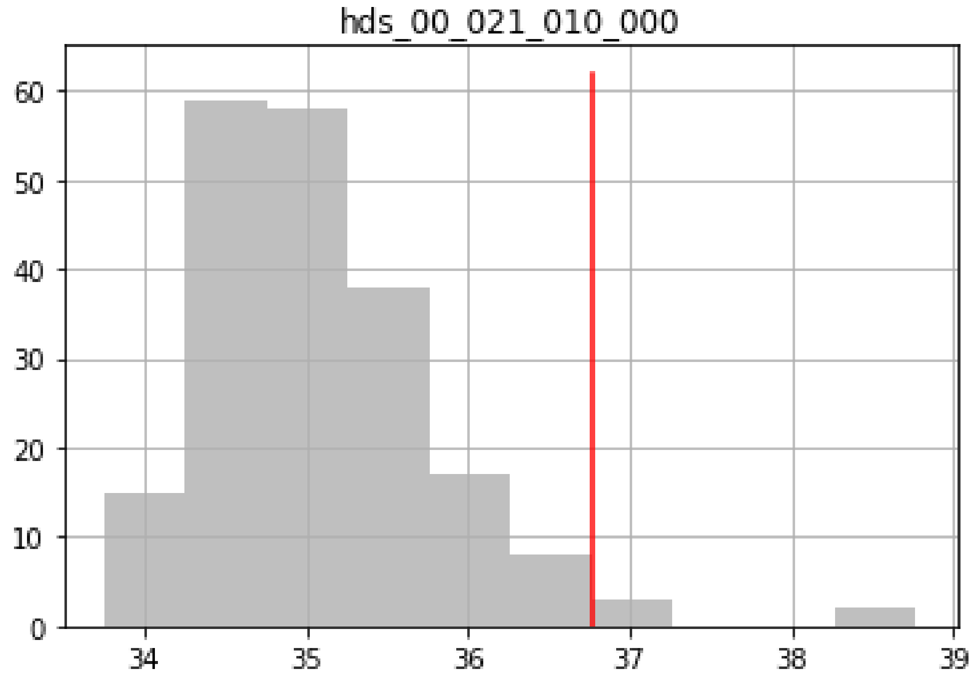

hds_00_002_015_000
50
40
30
20
10
0
34.50
34.75
35.00
35.25
35.50
35.75
36.00
36.25

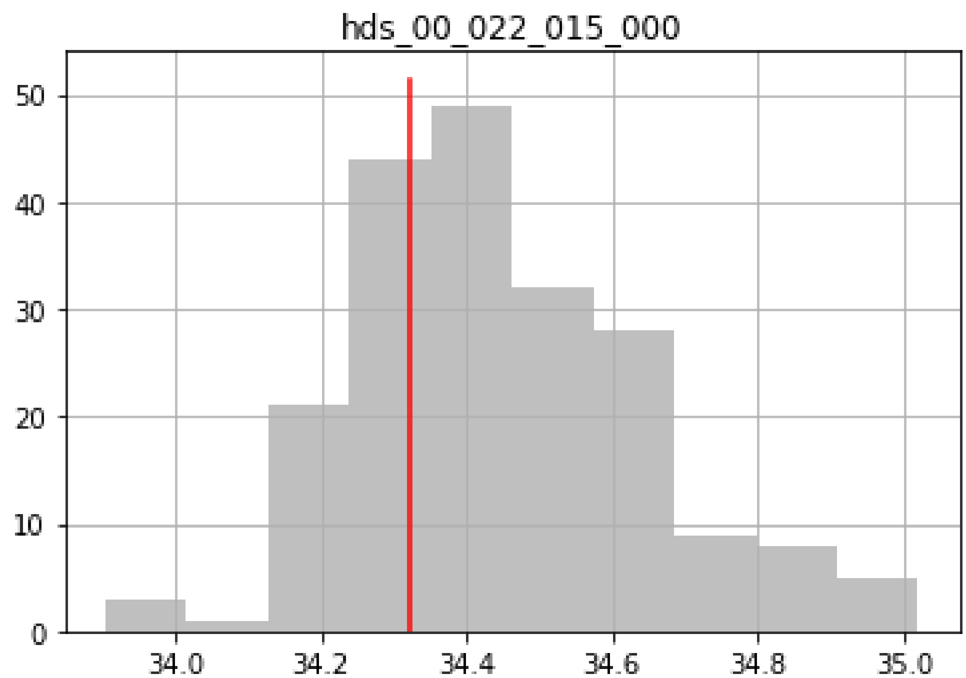

hds_00_003_008_000
0
10
20
30
40
50
36
38
40
42
44

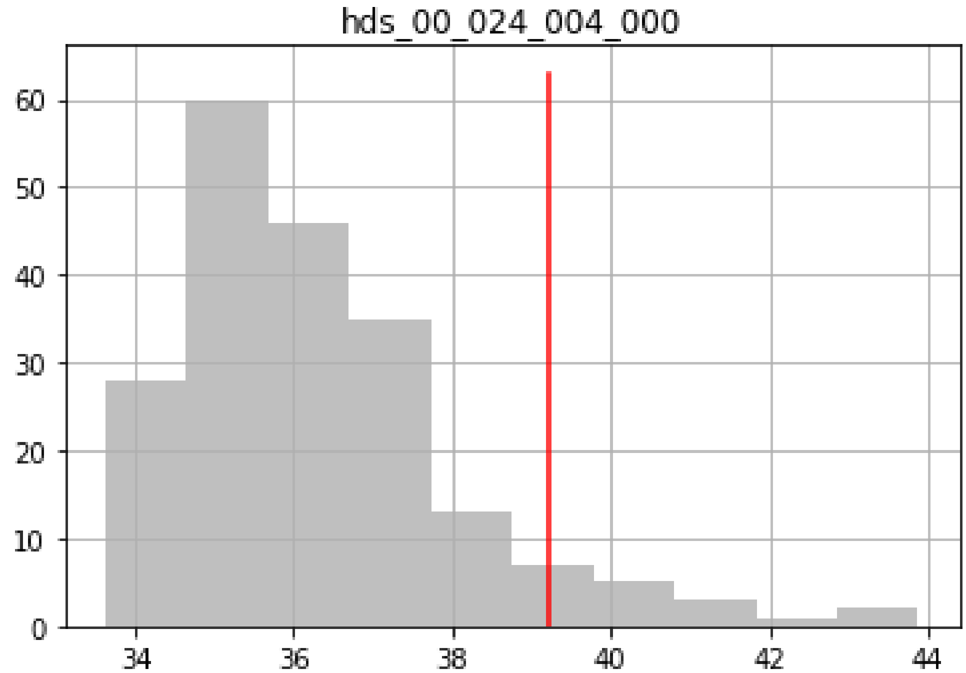

hds_00_009_001_000
0
10
20
30
40
50
35.0
37.5
40.0
42.5
45.0
47.5
50.0
52.5

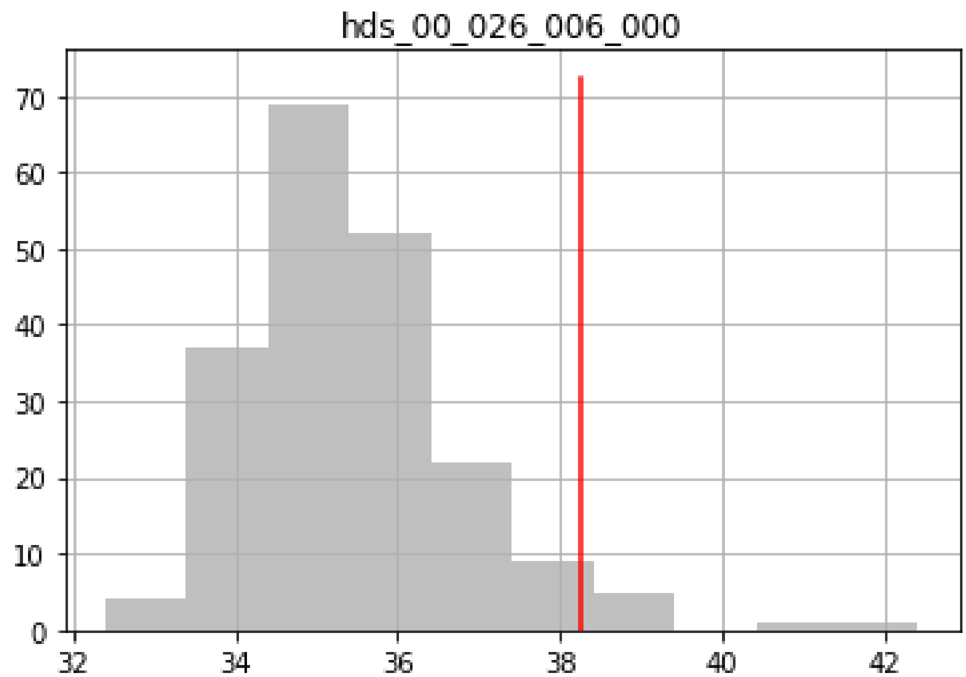

hds_00_013_010_000
0
10
20
30
40
34.0
34.5
35.0
35.5
36.0
36.5
37.0
37.5

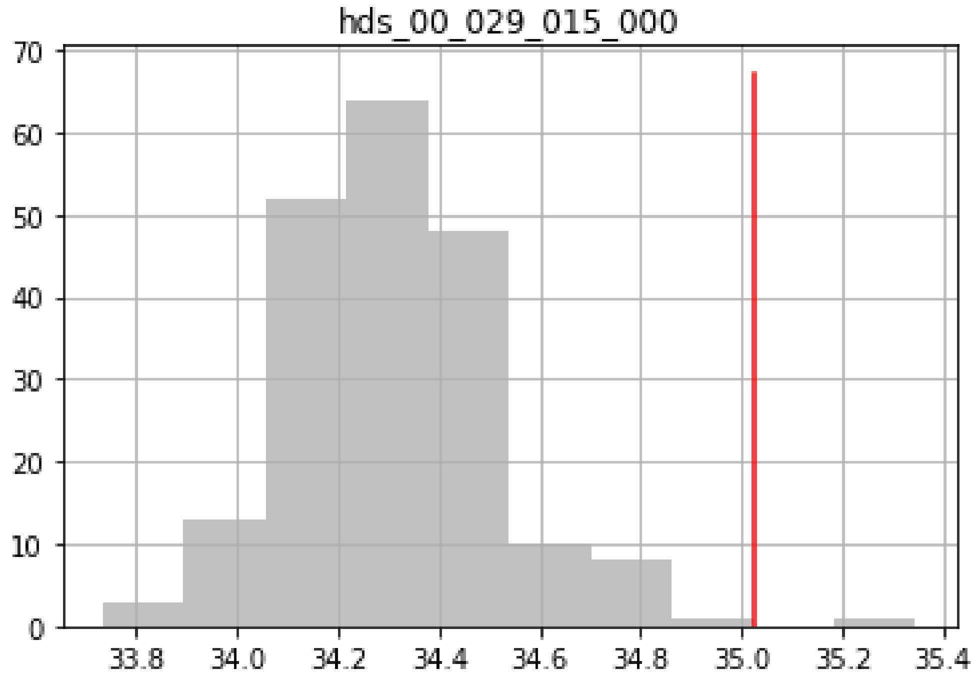

hds_00_015_016_000
0
10
20
30
40
50
34.0
34.2
34.4
34.6
34.8
35.0
35.2
35.4

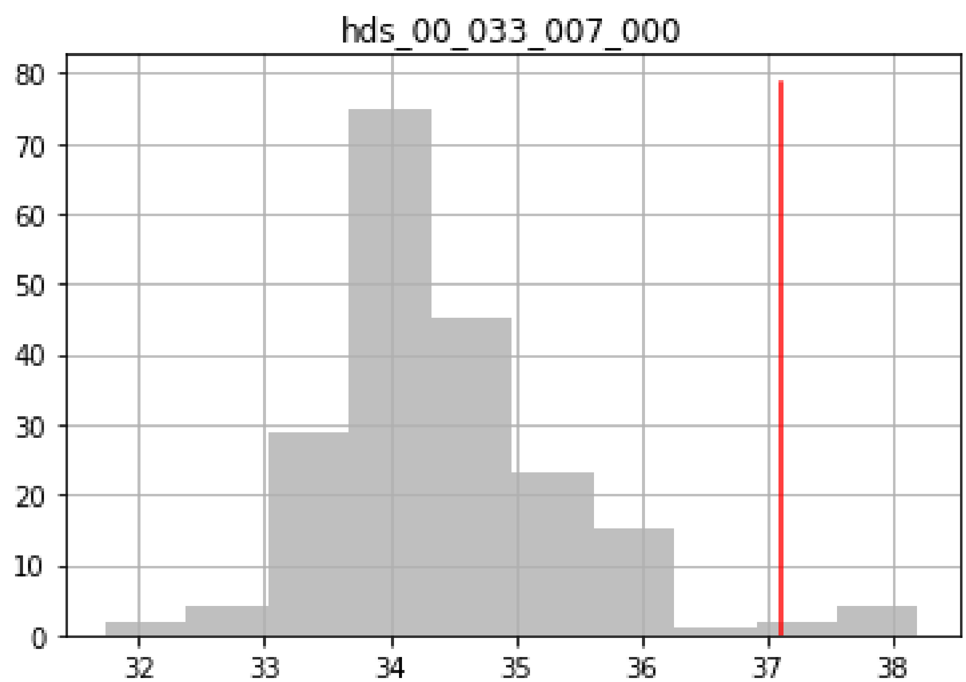

hds_00_021_010_000
60
50
40
30
20
10
0
34
35
36
37
38
39

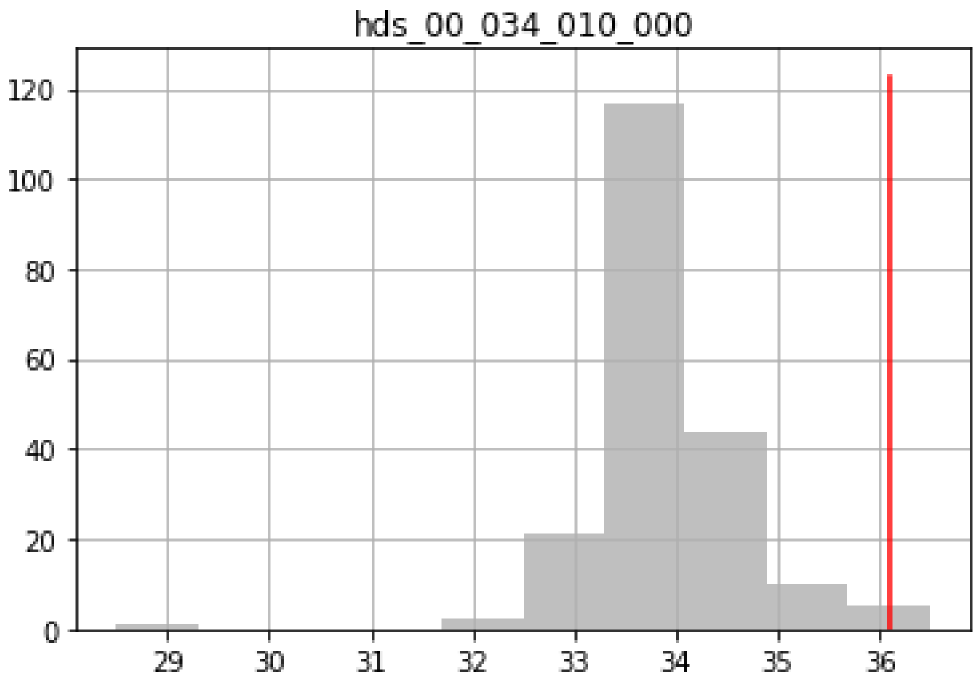

hds_00_022_015_000
50
40
30
20
10
0
34.0
34.2
34.4
34.6
34.8
35.0


hds_00_024_004_000
0
10
20
30
40
50
60
34
36
38
40
42
44


hds_00_026_006_000
0
10
20
30
40
50
60
70
32
34
36
38
40
42


hds_00_029_015_000
70
60
50
40
30
20
10
0
33.8
34.0
34.2
34.4
34.6
34.8
35.0
35.2
35.4


hds_00_033_007_000
80
70
60
50
40
30
20
10
0
32
33
34
35
36
37
38


hds_00_034_010_000
120
100
80
60
40
20
0
29
30
31
32
33
34
35
36