prior_montecarlo

May 2, 2019

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

```
In [1]: import os
        import shutil
        import numpy as np
        import pandas as pd
        import matplotlib.pyplot as plt
        import flopy
        import pyemu
```

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

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

1.0.1 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...
```

1.0.2 run the prior ensemble in parallel locally

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

1.0.3 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)
obs_df.shape
```

Out[6]:
```
(200, 4465)
```

drop any failed runs

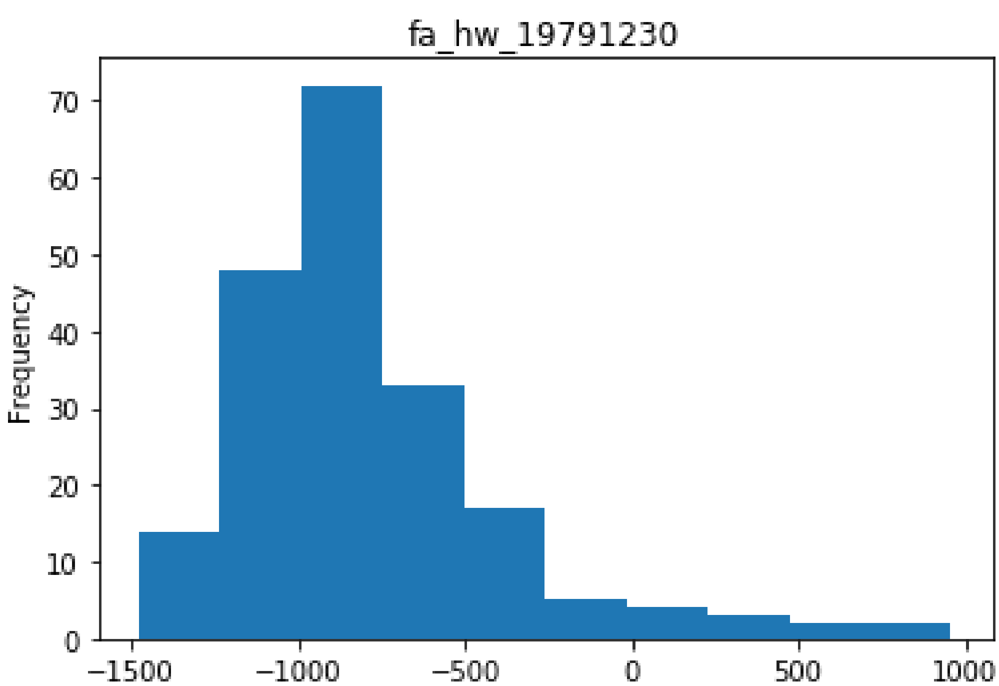
In [7]:
```
obs_df = obs_df.loc[obs_df.failed_flag==0,:]
obs_df.shape
```

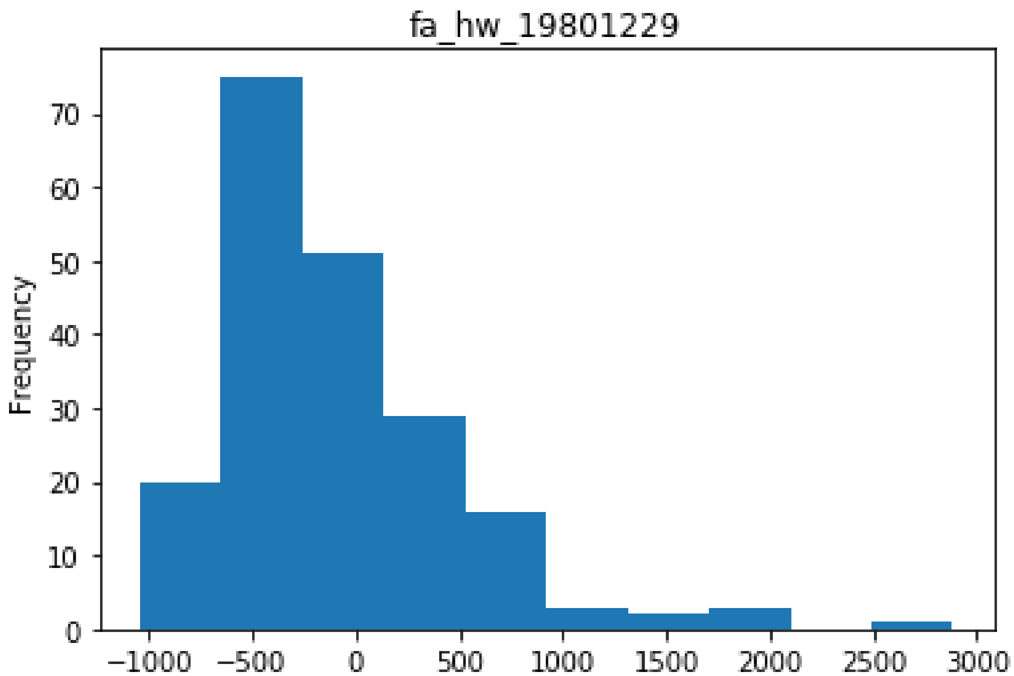
Out[7]:
```
(200, 4465)
```

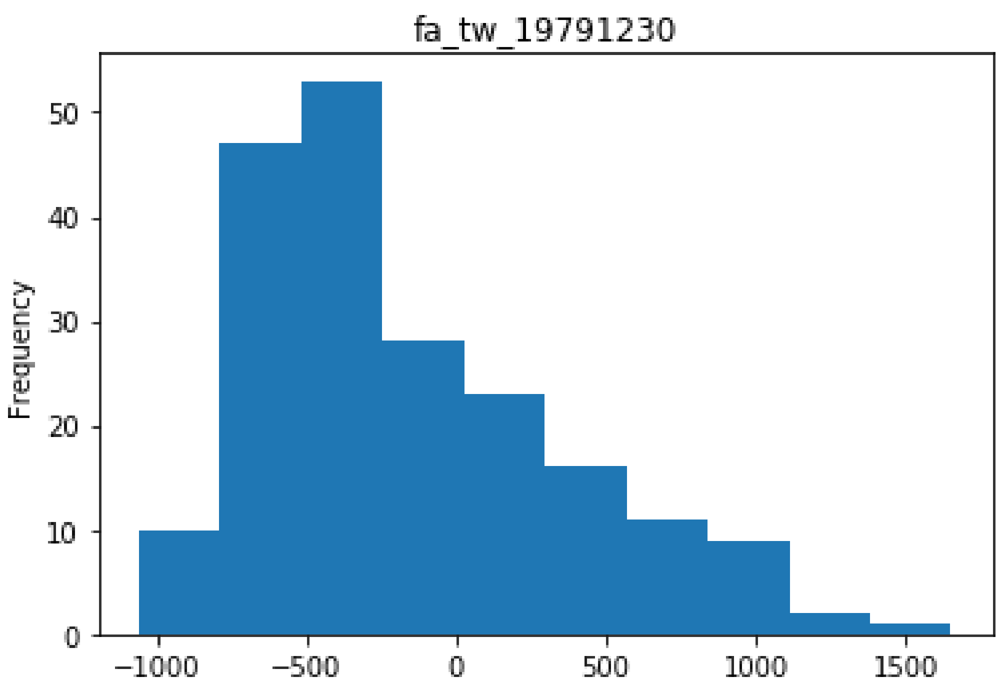
In [8]:
```
fnames = pst.pestpp_options["forecasts"].split(',')
fnames
```

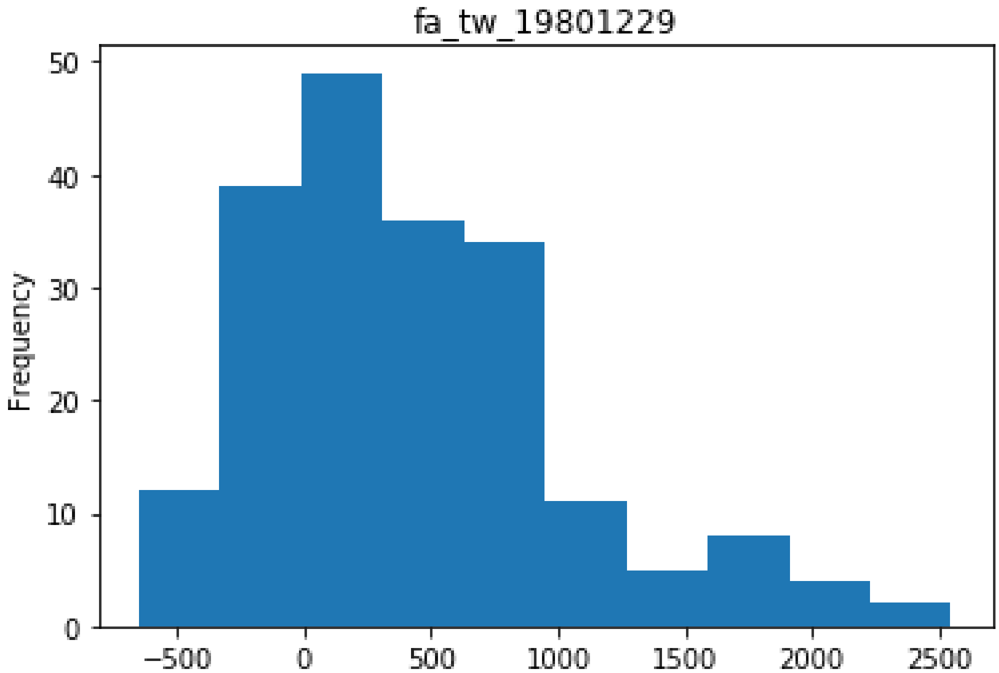
Out[8]:
```
['fa_hw_19791230',
 'fa_hw_19801229',
 'fa_tw_19791230',
 'fa_tw_19801229',
 'hds_00_013_002_000',
 'hds_00_013_002_001',
 'part_time',
 'part_status']
```

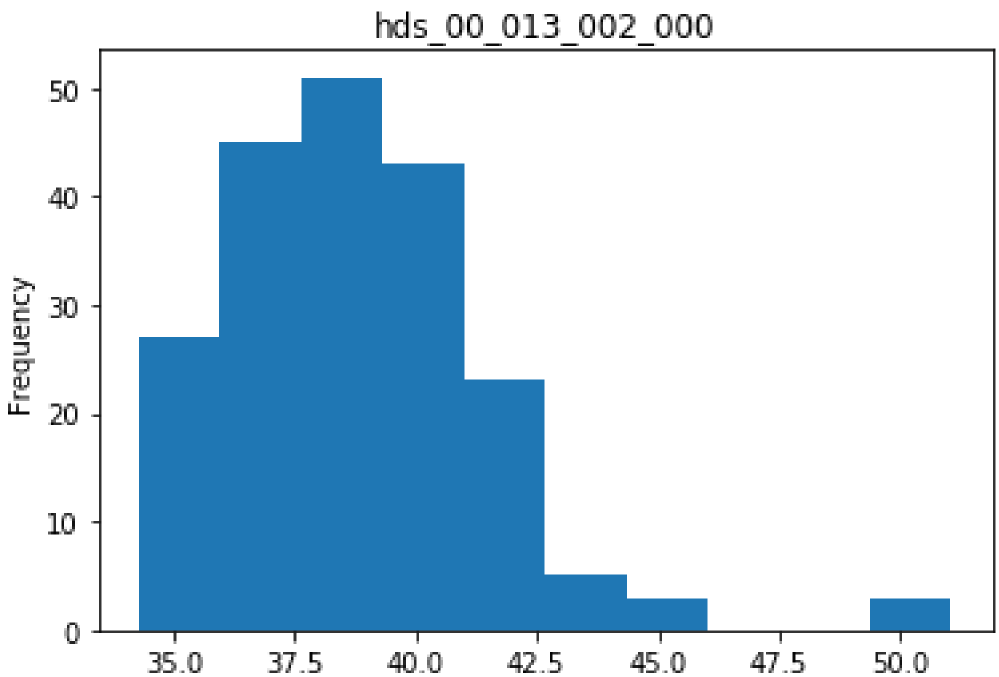
In [9]:
```
for forecast in fnames:
    ax = obs_df.loc[:,forecast].plot(kind="hist")
    ax.set_title(forecast)
    plt.show()
```

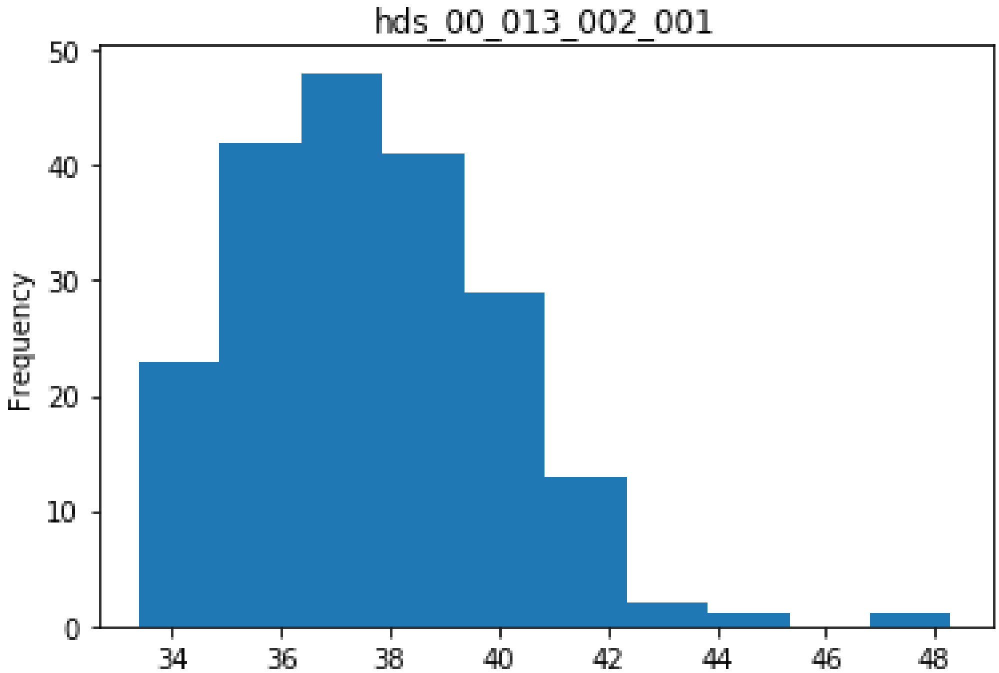


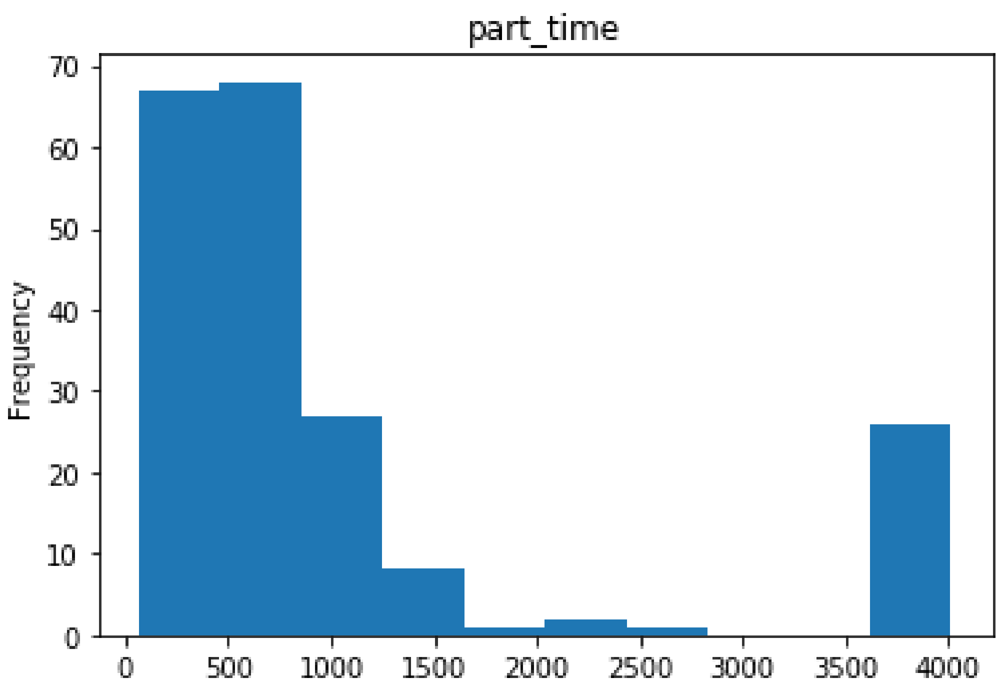

fa_hw_19801229
Frequency
0
10
20
30
40
50
60
70
−1000
−500
0
500
1000
1500
2000
2500
3000

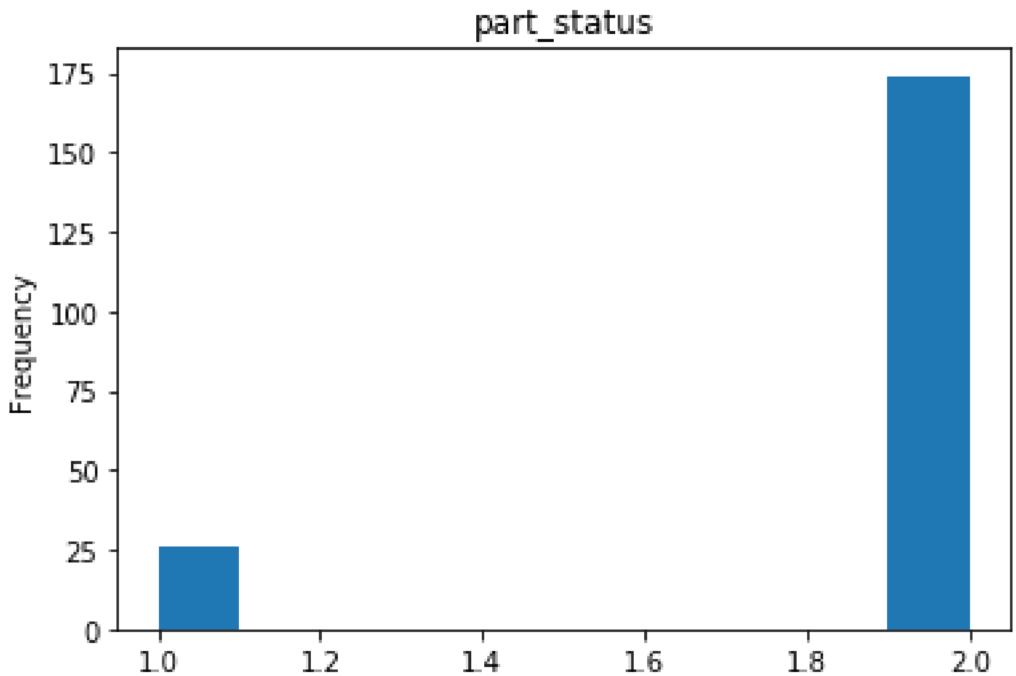

fa_tw_19791230
Frequency
0
10
20
30
40
50
−1000
−500
0
500
1000
1500


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


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


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

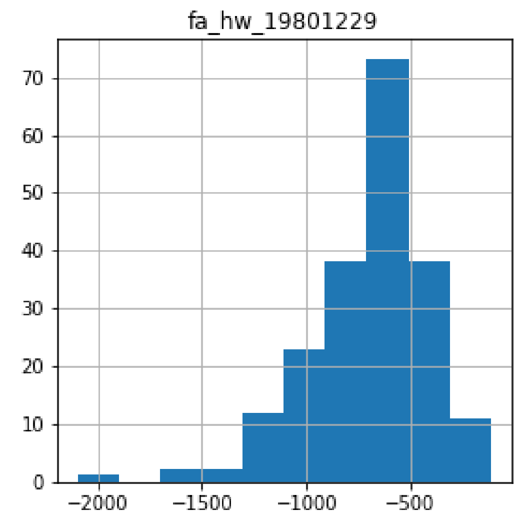

part_time
Frequency
0
10
20
30
40
50
60
70
0
500
1000
1500
2000
2500
3000
3500
4000

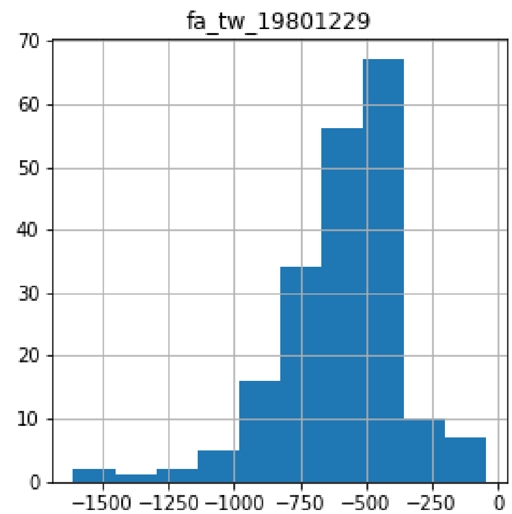


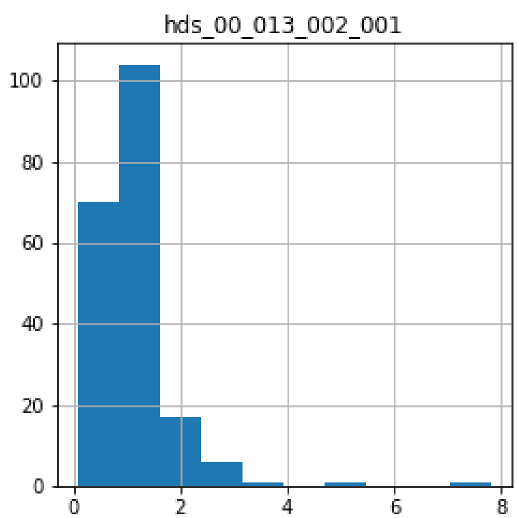
We see that under scenario conditions, many more realizations for the flow to the aquifer in the headwaters are postive (as expected). Lets difference these two:

```
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))
```

```
Out[10]: array([[<matplotlib.axes._subplots.AxesSubplot object at 0x108aeee10>,
                 <matplotlib.axes._subplots.AxesSubplot object at 0x181e6b5550>],
                [<matplotlib.axes._subplots.AxesSubplot object at 0x181e6ccac8>,
                 <matplotlib.axes._subplots.AxesSubplot object at 0x181e6ea080>]],
               dtype=object)
```







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

1.0.4 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. Im 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[20]
         idx
```

```
Out[11]: 56
```

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

```
Out[12]: fo_39_19791230        11545.000000
         hds_00_002_009_000       35.945004
         hds_00_002_015_000       34.993938
         hds_00_003_008_000       36.179363
         hds_00_009_001_000       38.799004
         hds_00_013_010_000       35.164818
         hds_00_015_016_000       34.681568
         hds_00_021_010_000       35.060108
         hds_00_022_015_000       34.568565
         hds_00_024_004_000       35.596191
         hds_00_026_006_000       34.910564
         hds_00_029_015_000       34.335033
         hds_00_033_007_000       34.046440
         hds_00_034_010_000       33.606953
         Name: 56, dtype: float64
```

Lets see how our selected truth does with the swgw forecasts:

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

```
Out[13]: fa_hw_19791230       -1187.493500
         fa_hw_19801229        -528.937500
         fa_tw_19791230         203.422730
         fa_tw_19801229         721.340850
         hds_00_013_002_000      39.006870
         hds_00_013_002_001      38.093163
         part_time              468.757400
         part_status              2.000000
         Name: 56, dtype: float64
```

```
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"] = 5.0
         obs.loc[obs.obgnme=="calflux","weight"] = 0.035
         obs.weight.value_counts()
```

```
Out[14]: 0.000    4422
         5.000      13
         0.035       1
         Name: weight, dtype: int64
```

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 [15]:
```python
# this should give the same standard normal draws each time
np.random.seed(seed=0)
snd = np.random.randn(pst.nnz_obs)
snd
```

Out[15]:
```
array([ 1.76405235,  0.40015721,  0.97873798,  2.2408932 ,  1.86755799,
       -0.97727788,  0.95008842, -0.15135721, -0.10321885,  0.4105985 ,
        0.14404357,  1.45427351,  0.76103773,  0.12167502])
```

In [16]:
```python
noise = snd * 1./obs.loc[pst.nnz_obs_names,"weight"]
noise
```

Out[16]:
```
obsnme
fo_39_19791230        50.401496
hds_00_002_009_000     0.080031
hds_00_002_015_000     0.195748
hds_00_003_008_000     0.448179
hds_00_009_001_000     0.373512
hds_00_013_010_000    -0.195456
hds_00_015_016_000     0.190018
hds_00_021_010_000    -0.030271
hds_00_022_015_000    -0.020644
hds_00_024_004_000     0.082120
hds_00_026_006_000     0.028809
hds_00_029_015_000     0.290855
hds_00_033_007_000     0.152208
hds_00_034_010_000     0.024335
Name: weight, dtype: float64
```

Only run this block once!!!

In [17]:
```python
pst.observation_data.loc[noise.index,"obsval"] += noise
pst.write(os.path.join(t_d,"freyberg.pst"))
pyemu.os_utils.run("pestpp-ies freyberg.pst",cwd=t_d)
```

In [18]:
```python
pst = pyemu.Pst(os.path.join(t_d,"freyberg.pst"))
print(pst.phi)
pst.res.loc[pst.nnz_obs_names,:]
```

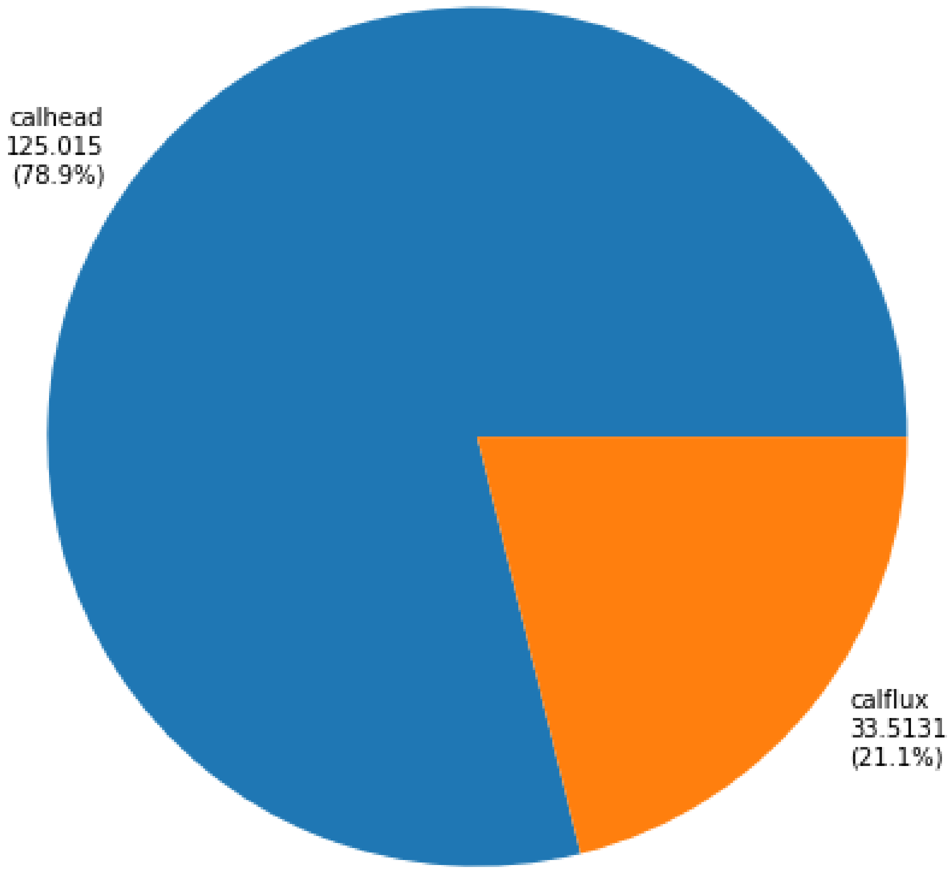
```
158.52813633665653
```

Out[18]:
```
                                  name    group      measured      modelled  \
name
fo_39_19791230          fo_39_19791230  calflux  11595.401496  11430.000000
hds_00_002_009_000  hds_00_002_009_000  calhead     36.025035     37.107498
hds_00_002_015_000  hds_00_002_015_000  calhead     35.189686     35.045185
hds_00_003_008_000  hds_00_003_008_000  calhead     36.627542     37.397289
hds_00_009_001_000  hds_00_009_001_000  calhead     39.172515     39.546417
```

```
hds_00_013_010_000  hds_00_013_010_000  calhead     34.969362     35.571774
hds_00_015_016_000  hds_00_015_016_000  calhead     34.871586     34.835716
hds_00_021_010_000  hds_00_021_010_000  calhead     35.029837     35.386250
hds_00_022_015_000  hds_00_022_015_000  calhead     34.547922     34.577492
hds_00_024_004_000  hds_00_024_004_000  calhead     35.678311     36.760464
hds_00_026_006_000  hds_00_026_006_000  calhead     34.939373     35.896149
hds_00_029_015_000  hds_00_029_015_000  calhead     34.625888     34.453842
hds_00_033_007_000  hds_00_033_007_000  calhead     34.198648     34.678810
hds_00_034_010_000  hds_00_034_010_000  calhead     33.631288     34.118073

                      residual  weight
name
fo_39_19791230      165.401496   0.035
hds_00_002_009_000   -1.082463   5.000
hds_00_002_015_000    0.144501   5.000
hds_00_003_008_000   -0.769747   5.000
hds_00_009_001_000   -0.373902   5.000
hds_00_013_010_000   -0.602411   5.000
hds_00_015_016_000    0.035870   5.000
hds_00_021_010_000   -0.356413   5.000
hds_00_022_015_000   -0.029570   5.000
hds_00_024_004_000   -1.082153   5.000
hds_00_026_006_000   -0.956776   5.000
hds_00_029_015_000    0.172046   5.000
hds_00_033_007_000   -0.480162   5.000
hds_00_034_010_000   -0.486785   5.000
```

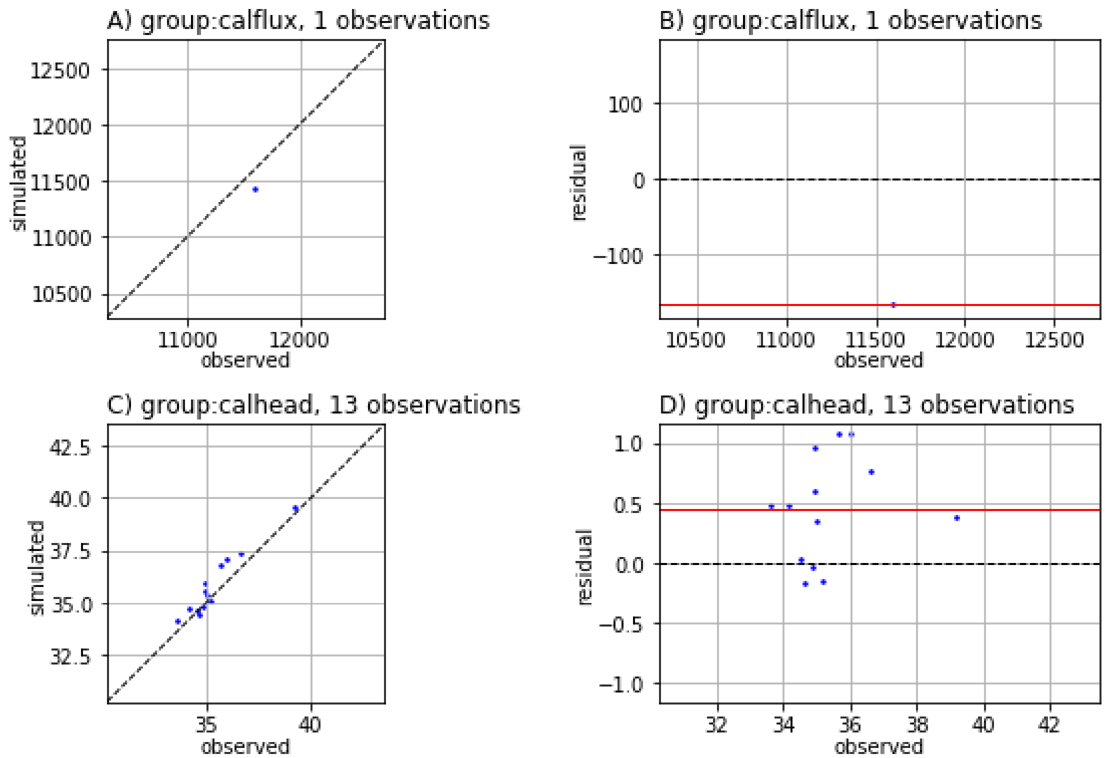
In [19]:
```
pst.phi_components
pst.plot(kind='phi_pie')
```

Out[19]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x181e590710>
```




```
In [20]: figs = pst.plot(kind="1to1")
```

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



Publication ready figs - oh snap!

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 [21]: 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"))
         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,:]
```

```
17.528847282834874
```

```
Out[21]:                                    name     group      measured      modelled  \
         name
         fo_39_19791230           fo_39_19791230  calflux  11595.401496  11545.000000
         hds_00_002_009_000  hds_00_002_009_000  calhead     36.025035     35.945004
         hds_00_002_015_000  hds_00_002_015_000  calhead     35.189686     34.993938
         hds_00_003_008_000  hds_00_003_008_000  calhead     36.627542     36.179363
         hds_00_009_001_000  hds_00_009_001_000  calhead     39.172515     38.799004
         hds_00_013_010_000  hds_00_013_010_000  calhead     34.969362     35.164818
         hds_00_015_016_000  hds_00_015_016_000  calhead     34.871586     34.681568
         hds_00_021_010_000  hds_00_021_010_000  calhead     35.029837     35.060108
         hds_00_022_015_000  hds_00_022_015_000  calhead     34.547922     34.568565
         hds_00_024_004_000  hds_00_024_004_000  calhead     35.678311     35.596191
         hds_00_026_006_000  hds_00_026_006_000  calhead     34.939373     34.910564
         hds_00_029_015_000  hds_00_029_015_000  calhead     34.625888     34.335033
         hds_00_033_007_000  hds_00_033_007_000  calhead     34.198648     34.046440
         hds_00_034_010_000  hds_00_034_010_000  calhead     33.631288     33.606953

                              residual  weight
         name
         fo_39_19791230      50.401496   0.035
         hds_00_002_009_000   0.080031   5.000
         hds_00_002_015_000   0.195748   5.000
         hds_00_003_008_000   0.448179   5.000
         hds_00_009_001_000   0.373512   5.000
         hds_00_013_010_000  -0.195456   5.000
         hds_00_015_016_000   0.190018   5.000
         hds_00_021_010_000  -0.030271   5.000
         hds_00_022_015_000  -0.020644   5.000
         hds_00_024_004_000   0.082120   5.000
         hds_00_026_006_000   0.028809   5.000
         hds_00_029_015_000   0.290855   5.000
         hds_00_033_007_000   0.152208   5.000
         hds_00_034_010_000   0.024335   5.000
```

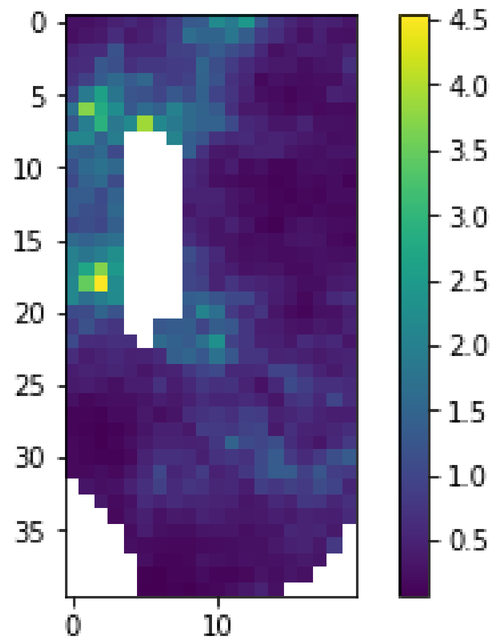
The residual should be exactly the noise values from above.Lets load the model (that was just run using the true pars) and check some things

```
In [22]: m = flopy.modflow.Modflow.load("freyberg.nam",model_ws=m_d)
```

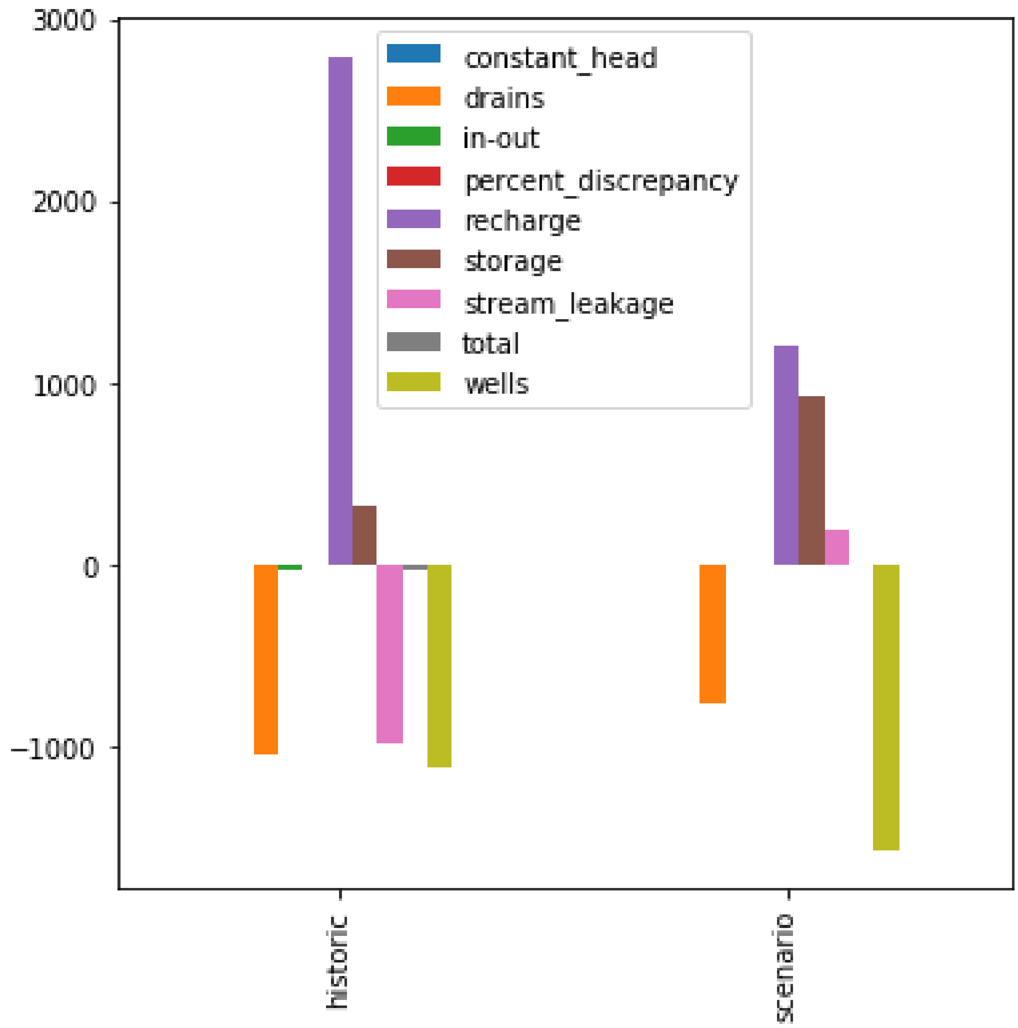
```
In [23]: a = m.upw.vka[1].array
         #a = m.rch.rech[0].array
         a = np.ma.masked_where(m.bas6.ibound[0].array==0,a)
         print(a.min(),a.max())
         c = plt.imshow(a)
         plt.colorbar(c)
```

```
0.07383982 4.534729
```

```
Out[23]: <matplotlib.colorbar.Colorbar at 0x181ef735f8>
```



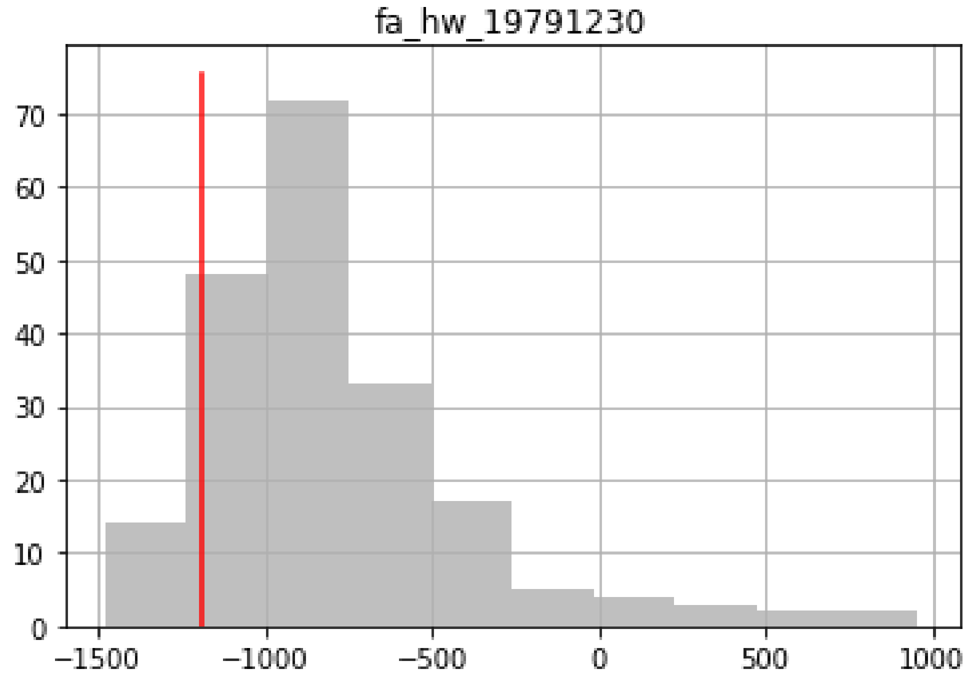

```
In [24]: lst = flopy.utils.MfListBudget(os.path.join(m_d,"freyberg.list"))
         df = lst.get_dataframes(diff=True)[0]
         ax = df.plot(kind="bar",figsize=(6,6))
         a = ax.set_xticklabels(["historic","scenario"],rotation=90)
```

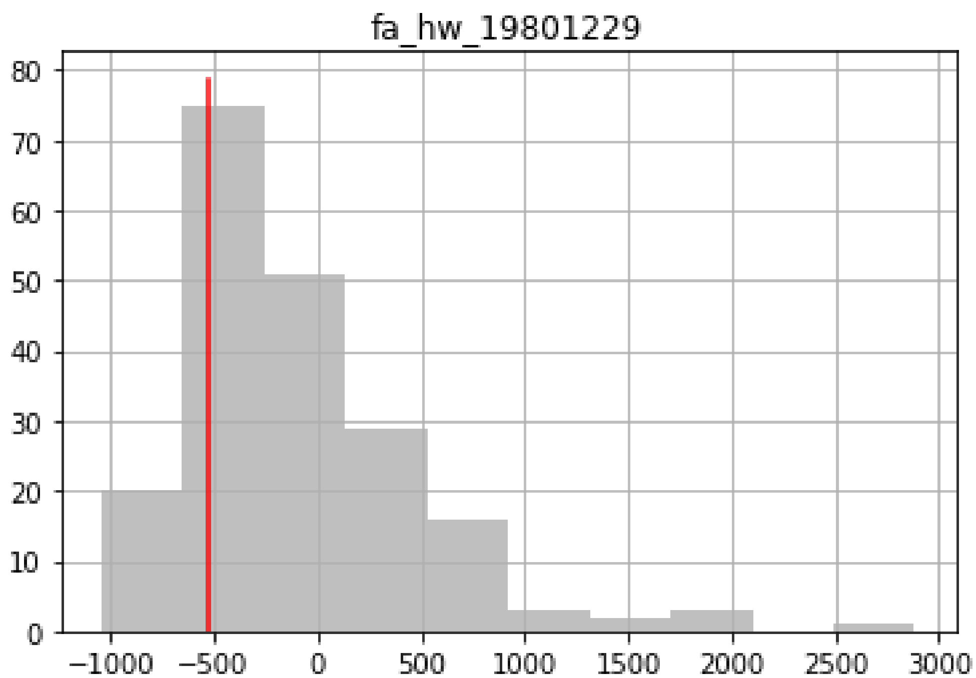


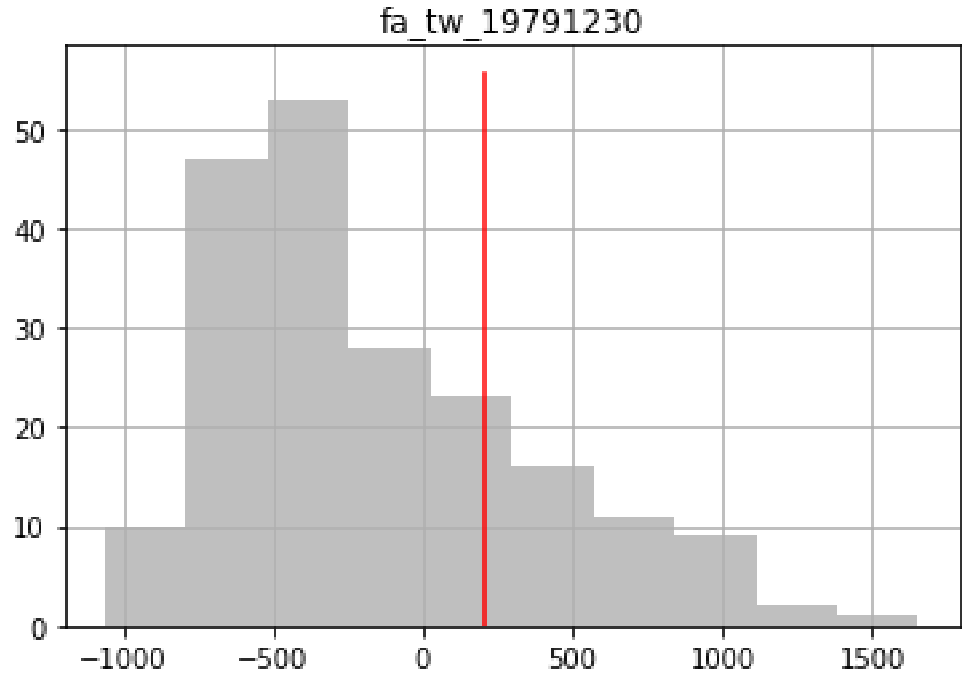
1.0.5 see how our existing observation ensemble compares to the truth

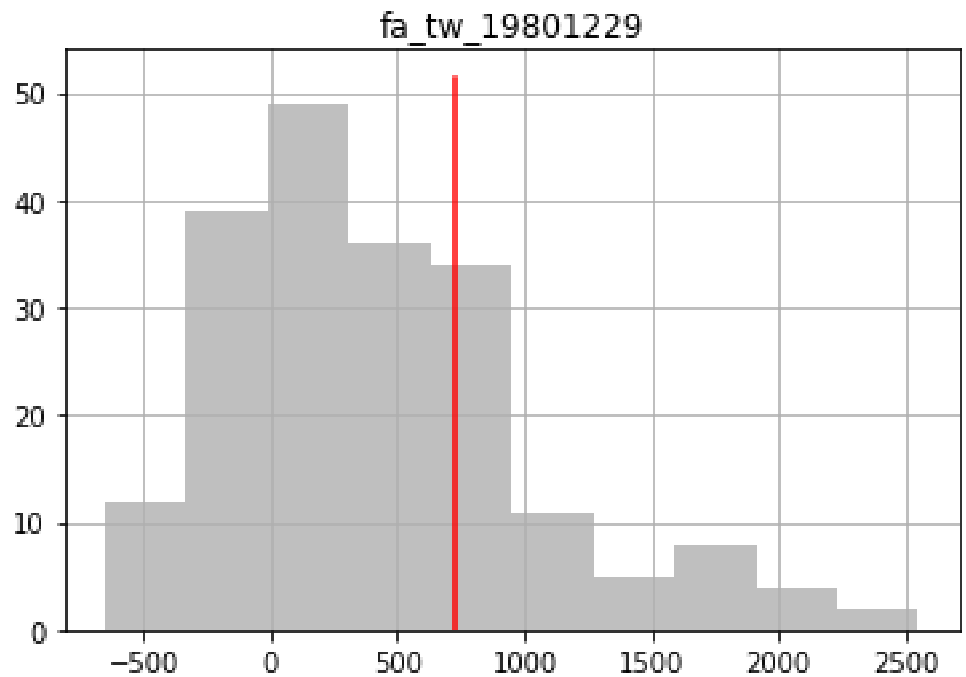
sw-gw outputs:

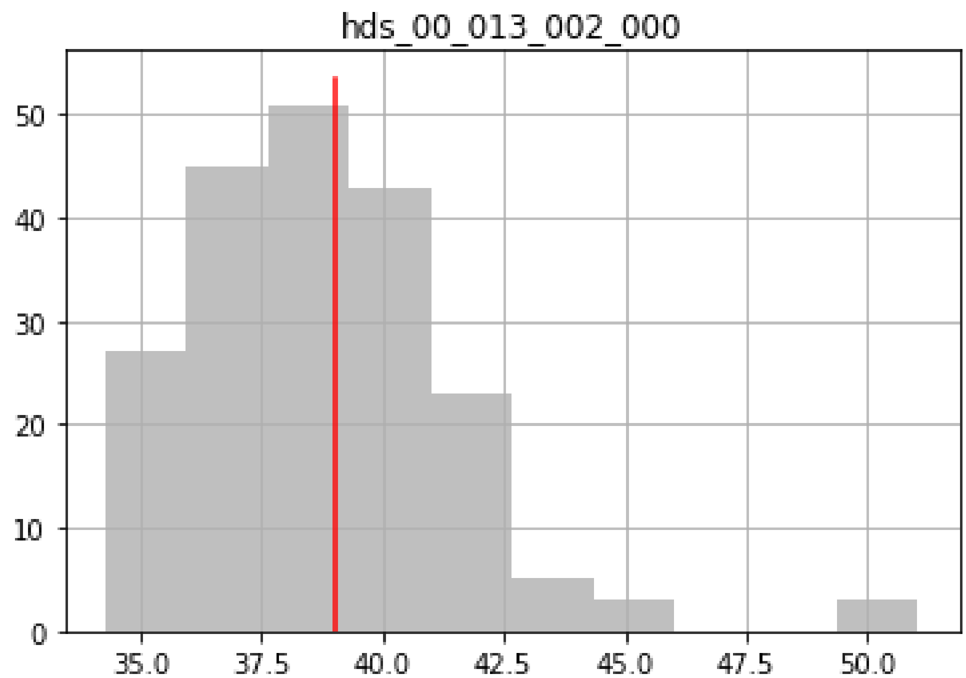
```
In [25]: obs = pst.observation_data

         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()
```

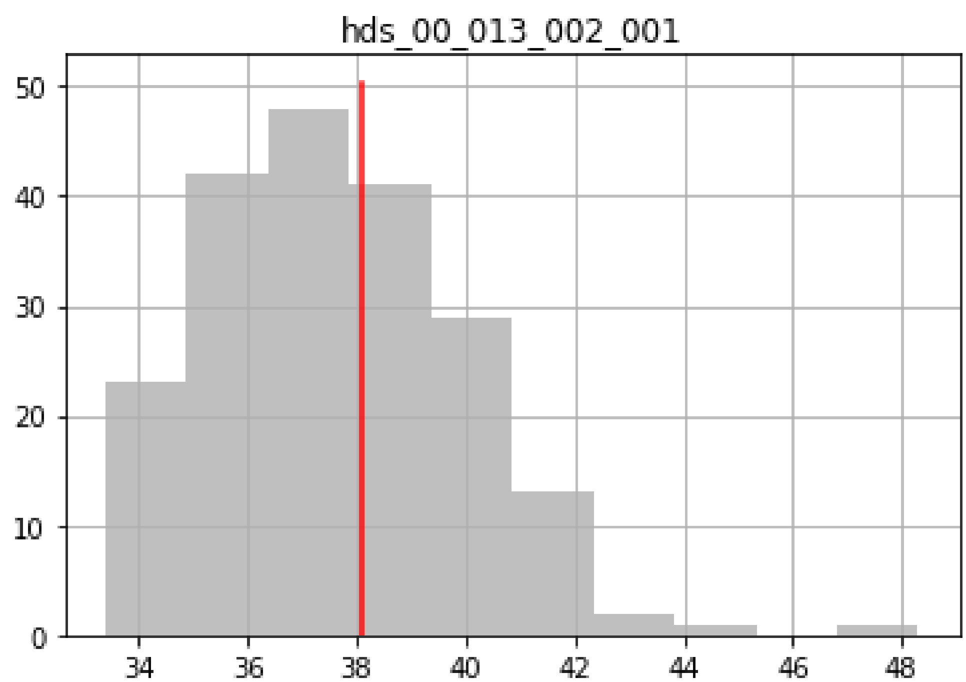

fa_hw_19791230
70
60
50
40
30
20
10
0
−1500
−1000
−500
0
500
1000

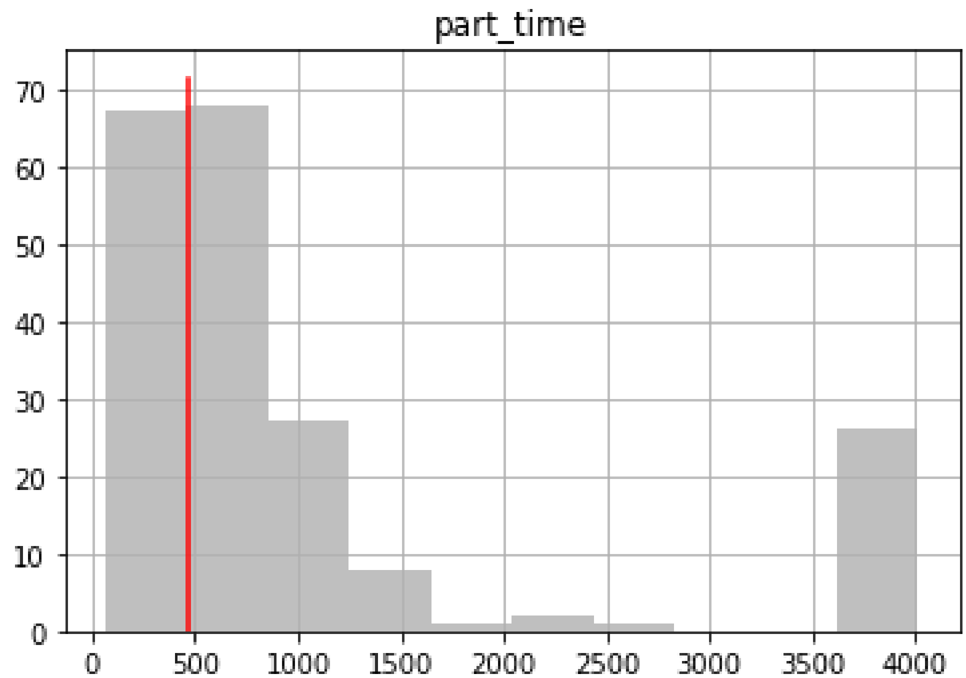

fa_hw_19801229
80
70
60
50
40
30
20
10
0
−1000
−500
0
500
1000
1500
2000
2500
3000

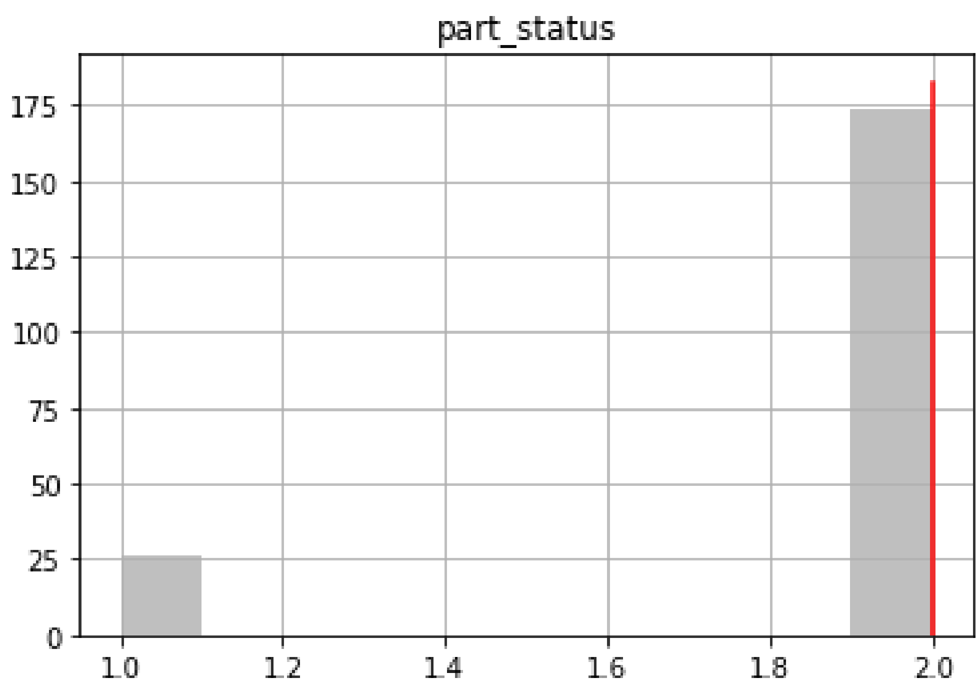

fa_tw_19791230
0
10
20
30
40
50
−1000
−500
0
500
1000
1500


fa_tw_19801229
0
10
20
30
40
50
−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

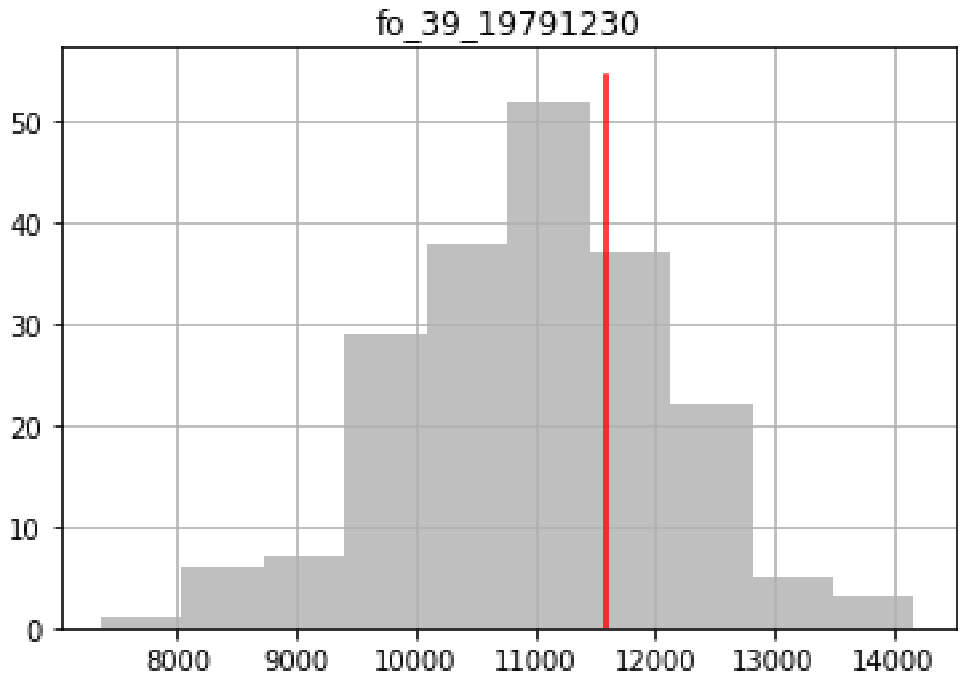

hds_00_013_002_001
0
10
20
30
40
50
34
36
38
40
42
44
46
48

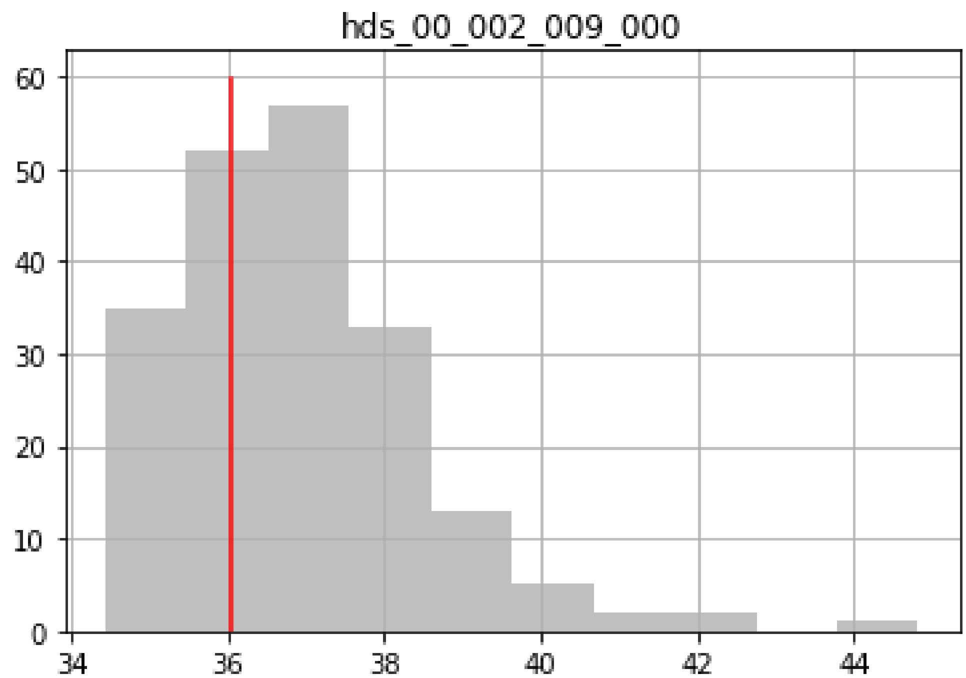


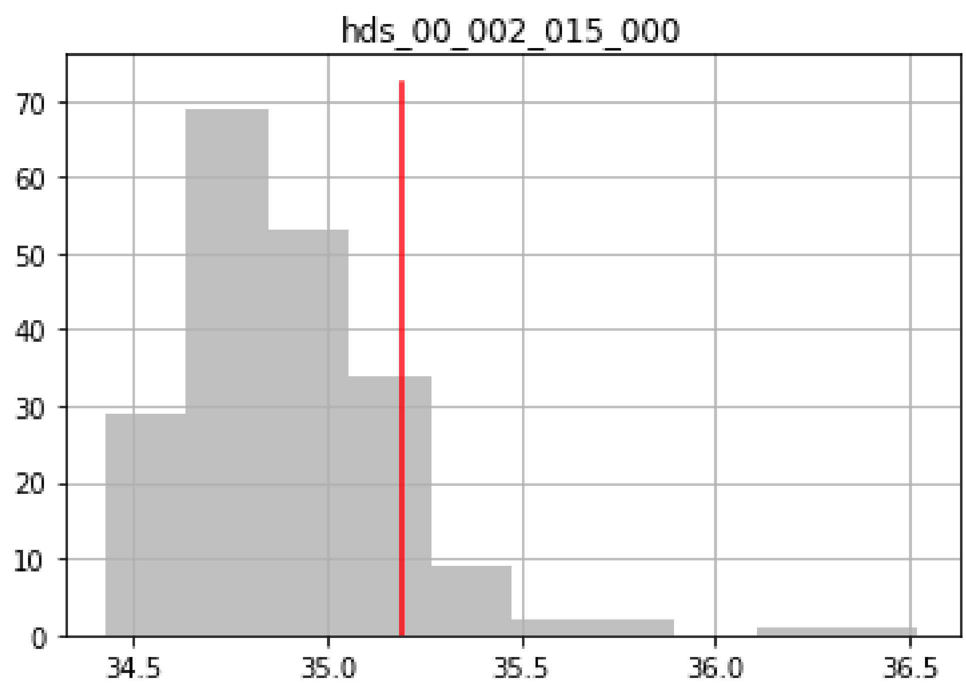


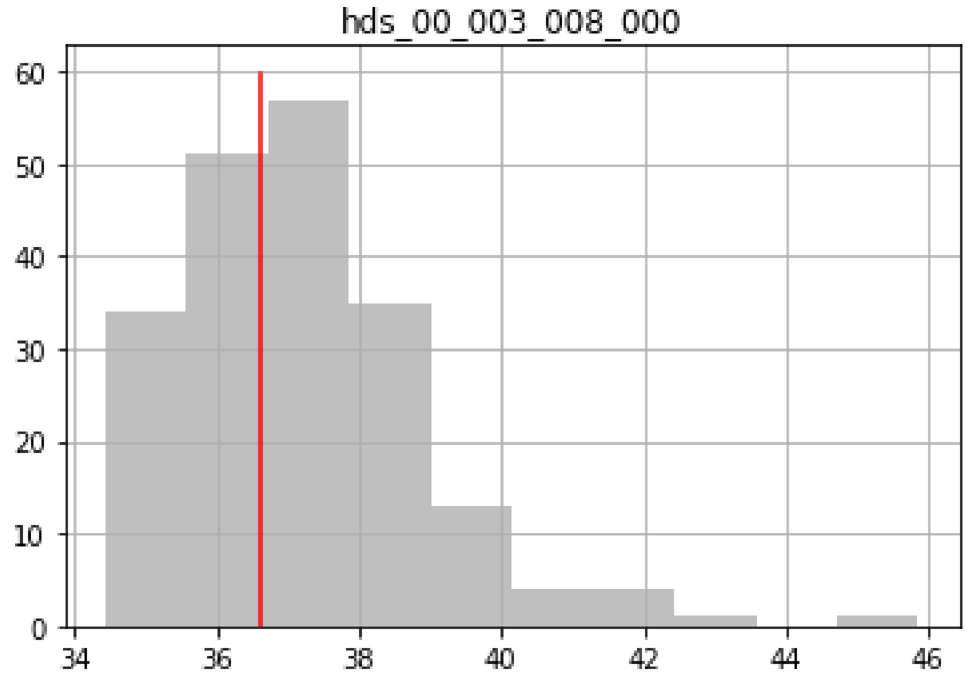
observations:

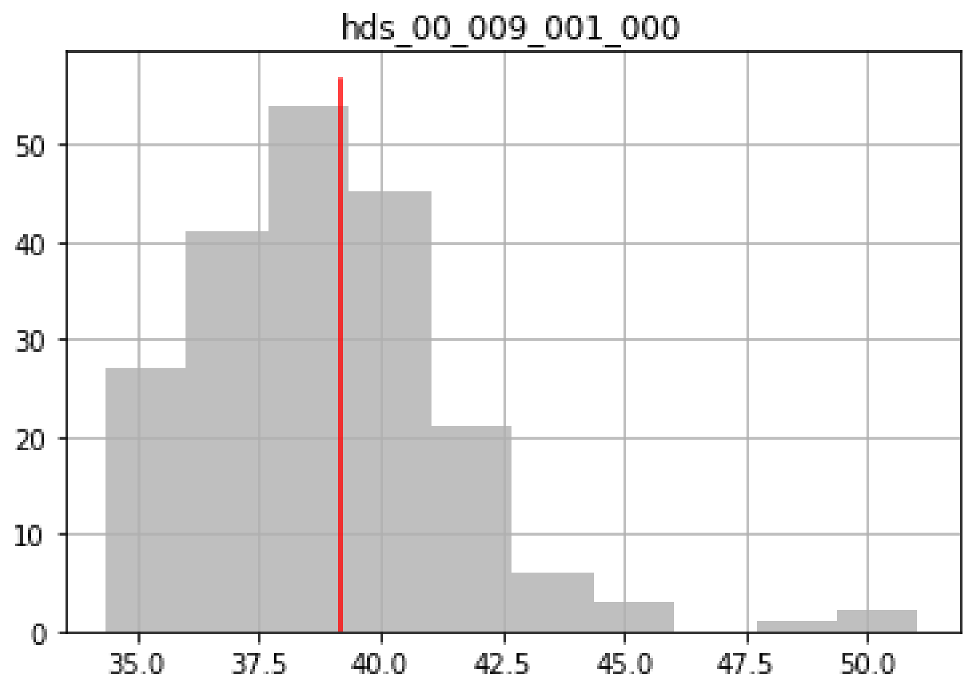
In [26]:
```
for oname in pst.nnz_obs_names:
    ax = plt.subplot(111)
    obs_df.loc[:,oname].hist(ax=ax,color="0.5",alpha=0.5)
    ax.plot([obs.loc[oname,"obsval"],obs.loc[oname,"obsval"]],ax.get_ylim(),"r")
    ax.set_title(oname)
    plt.show()
```

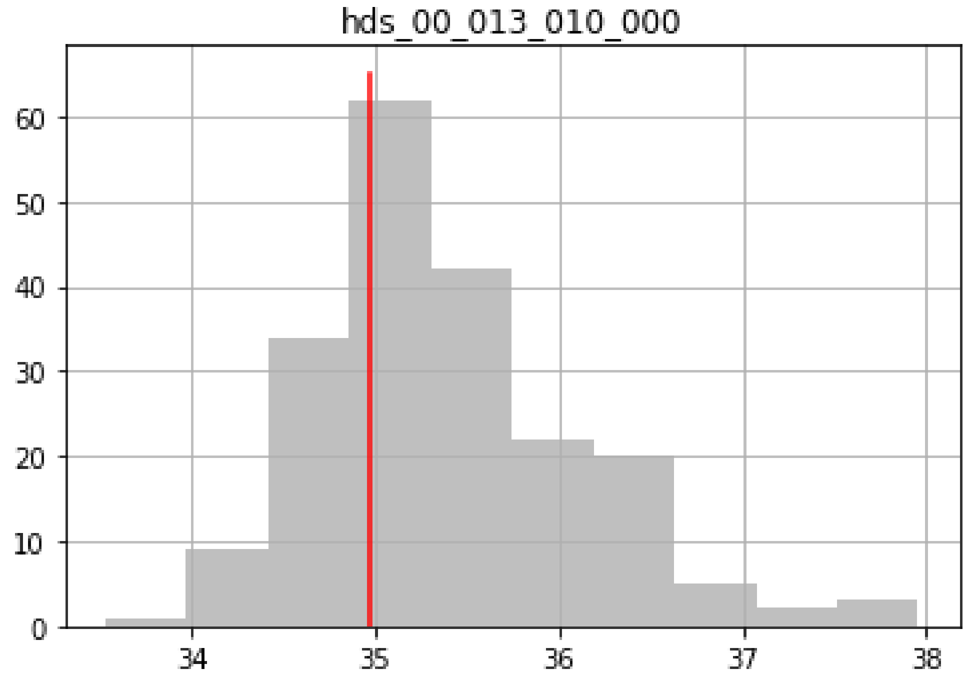


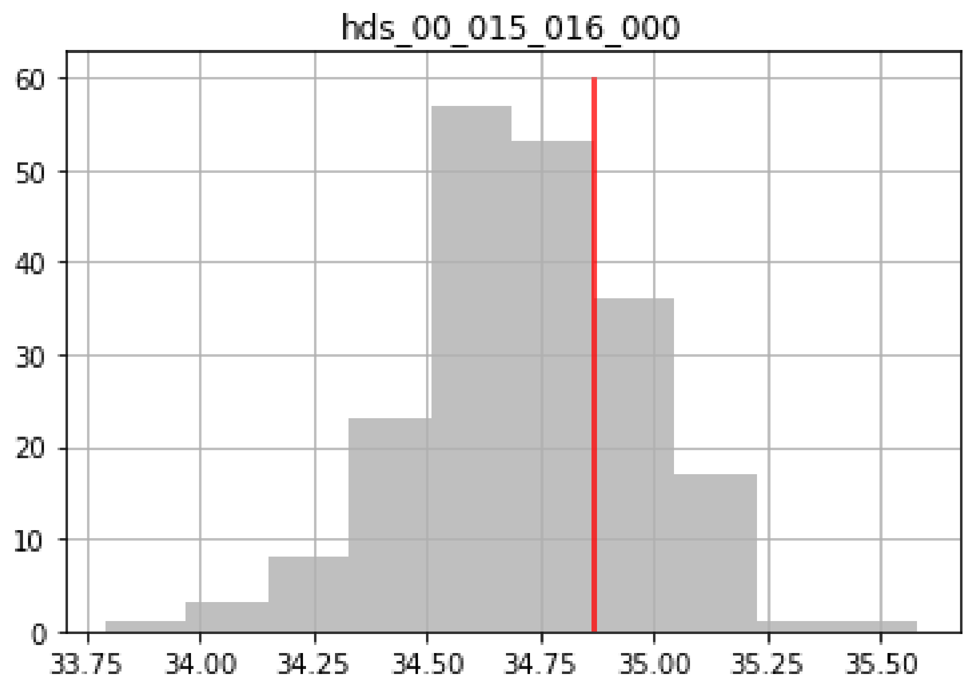

hds_00_002_009_000
60
50
40
30
20
10
0
34
36
38
40
42
44

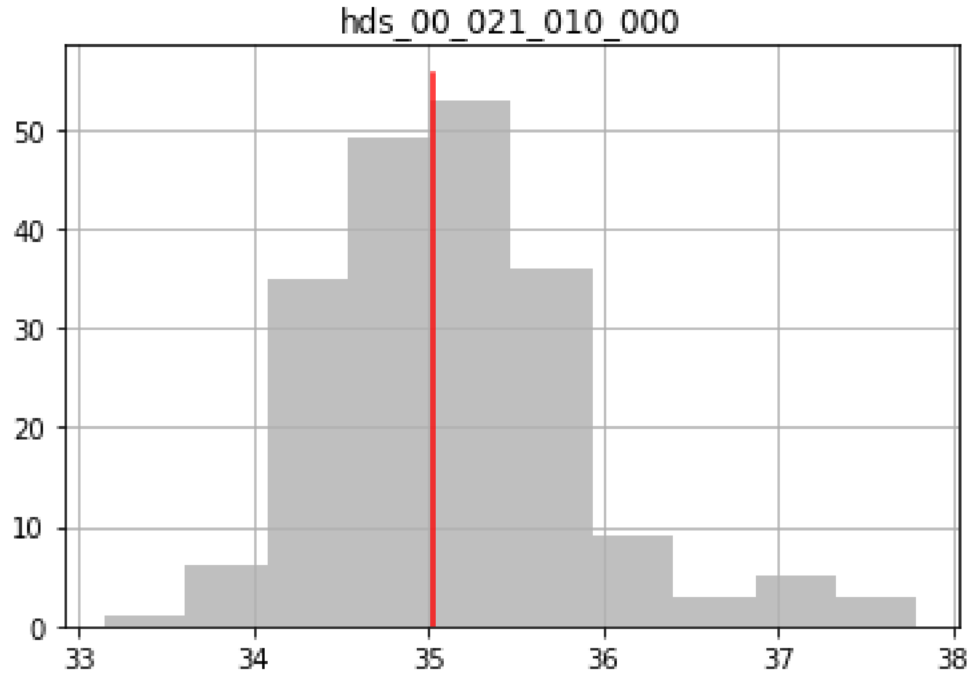

hds_00_002_015_000
70
60
50
40
30
20
10
0
34.5
35.0
35.5
36.0
36.5

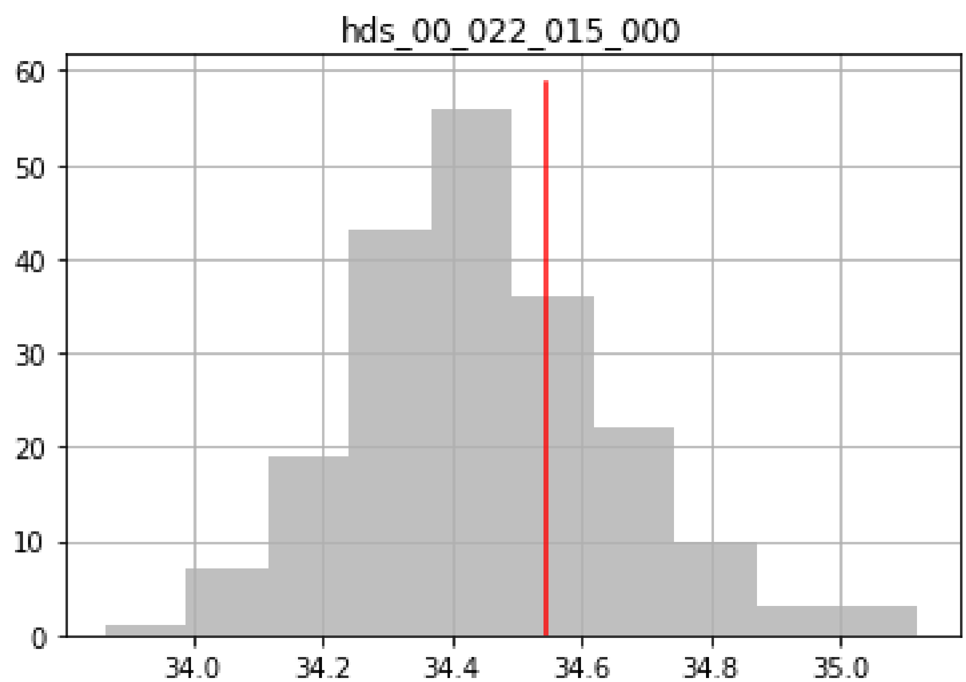

hds_00_003_008_000
0
10
20
30
40
50
60
34
36
38
40
42
44
46

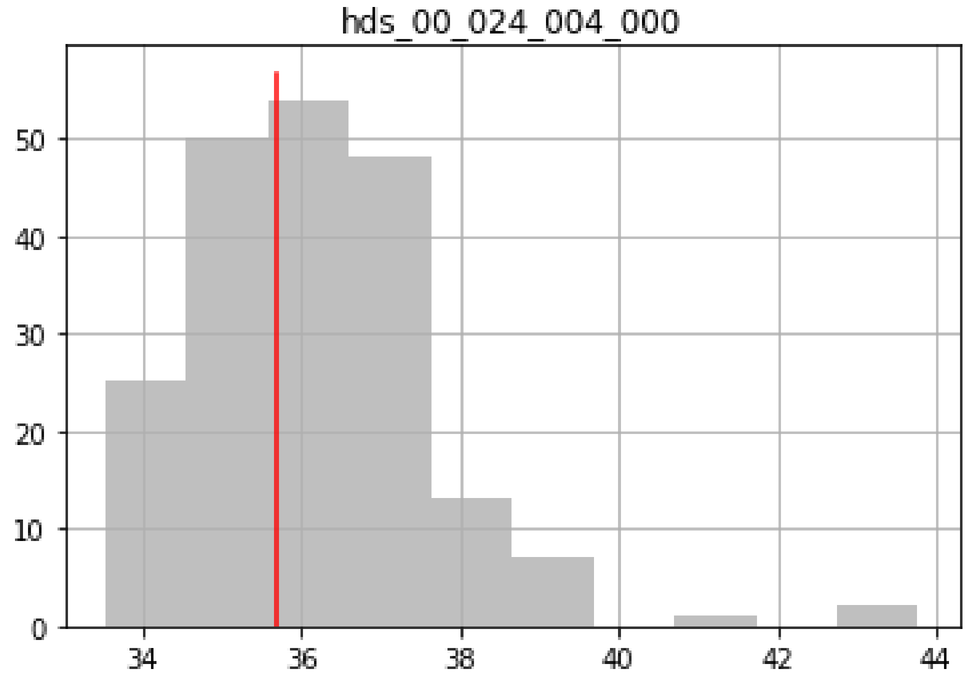

hds_00_009_001_000
0
10
20
30
40
50
35.0
37.5
40.0
42.5
45.0
47.5
50.0

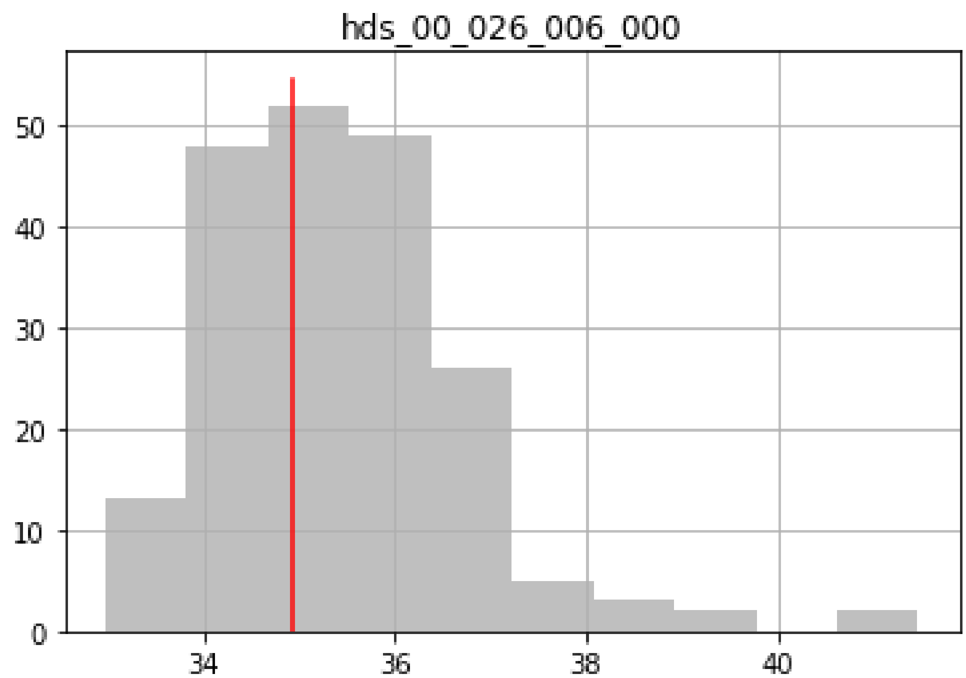

hds_00_013_010_000
60
50
40
30
20
10
0
34
35
36
37
38

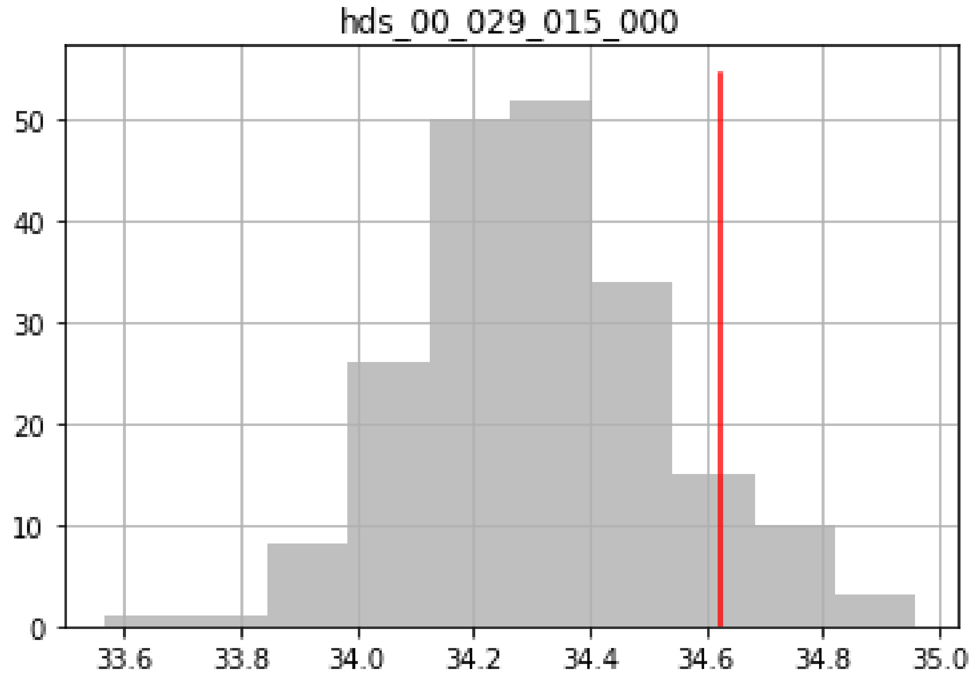

hds_00_015_016_000
60
50
40
30
20
10
0
33.75
34.00
34.25
34.50
34.75
35.00
35.25
35.50

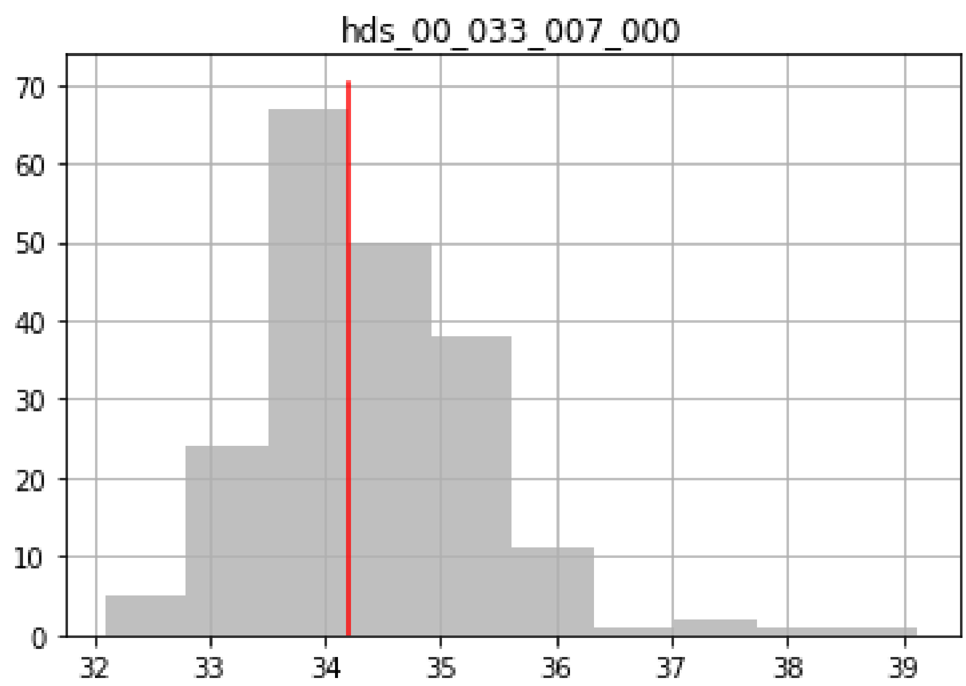

hds_00_021_010_000
0
10
20
30
40
50
33
34
35
36
37
38

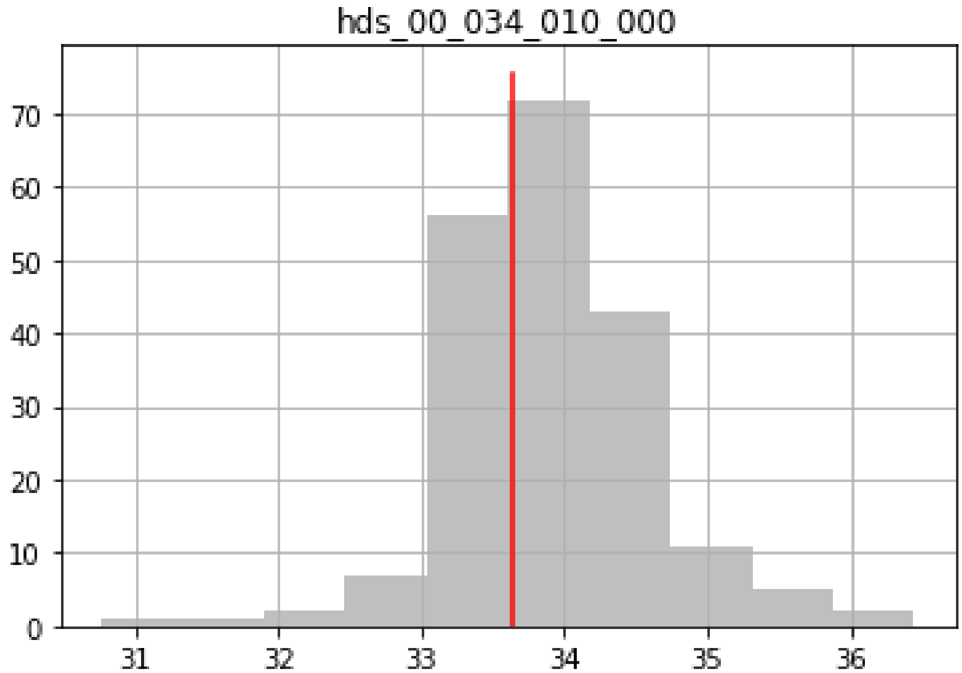

hds_00_022_015_000
0
10
20
30
40
50
60
34.0
34.2
34.4
34.6
34.8
35.0


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


hds_00_026_006_000
50
40
30
20
10
0
34
36
38
40


hds_00_029_015_000
0
10
20
30
40
50
33.6
33.8
34.0
34.2
34.4
34.6
34.8
35.0


hds_00_033_007_000
0
10
20
30
40
50
60
70
32
33
34
35
36
37
38
39


hds_00_034_010_000
0
10
20
30
40
50
60
70
31
32
33
34
35
36