prior_montecarlo

May 3, 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

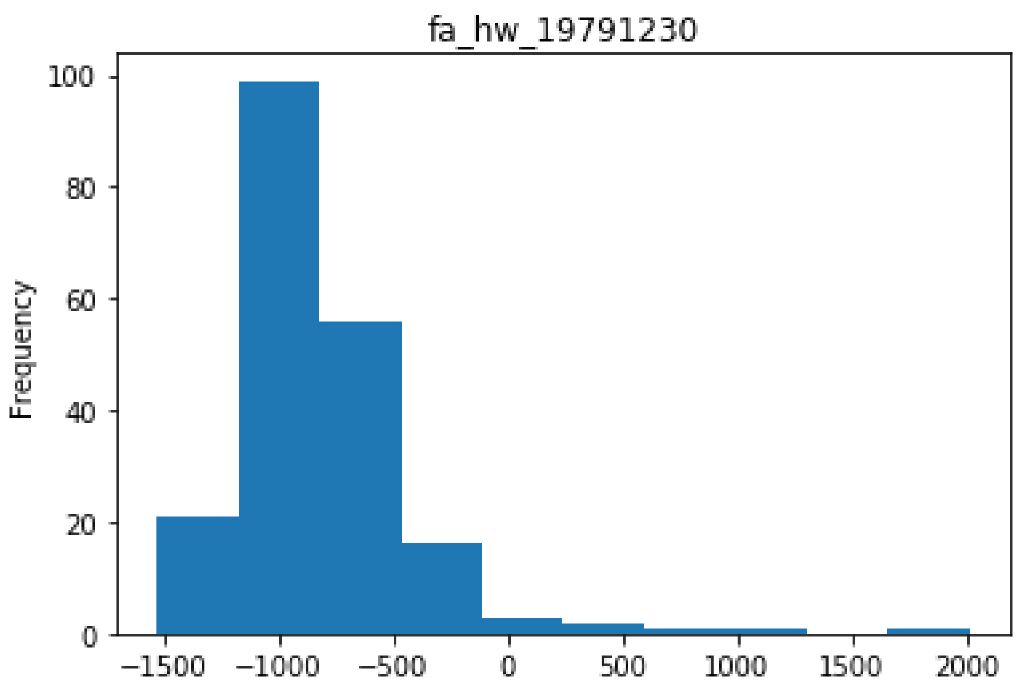
```
In [7]: obs_df = obs_df.loc[obs_df.failed_flag==0,:]
        obs_df.shape
```

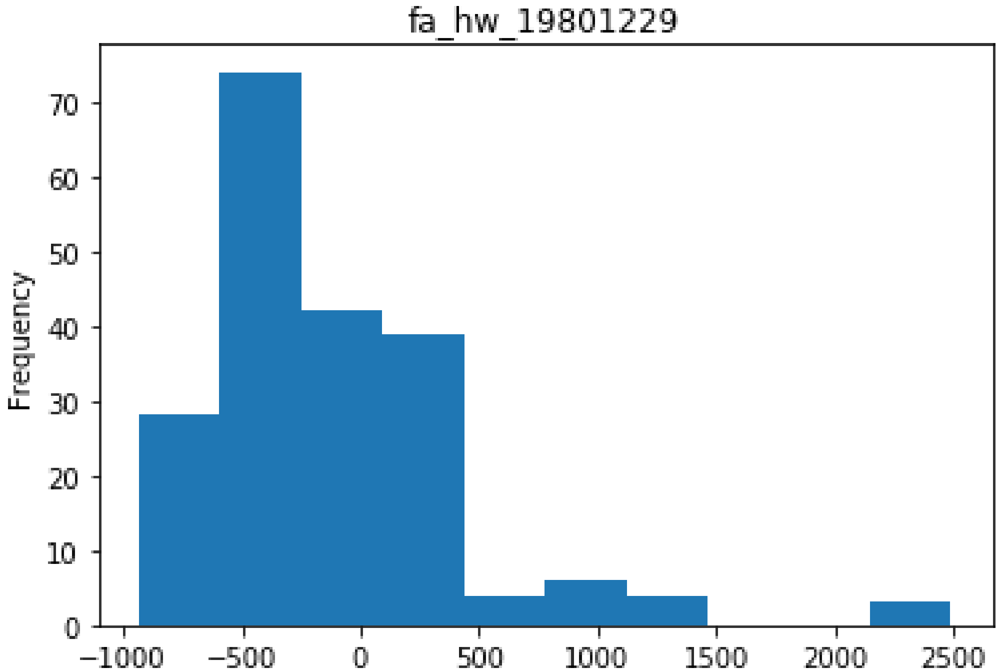
```
Out[7]: (200, 4465)
```

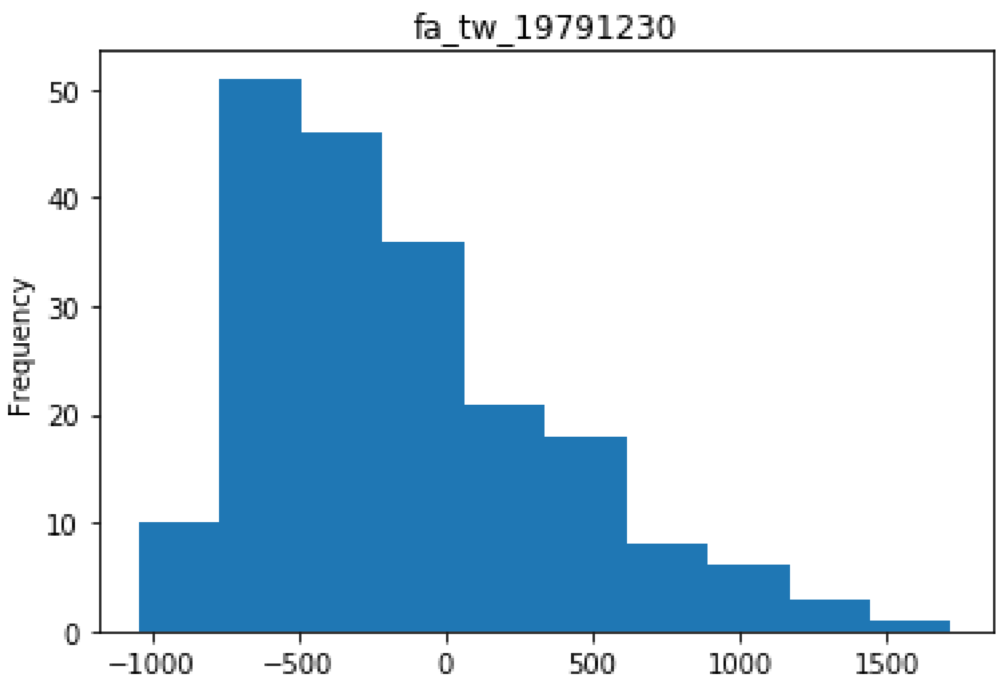
```
In [8]: fnames = pst.pestpp_options["forecasts"].split(',')
        fnames
```

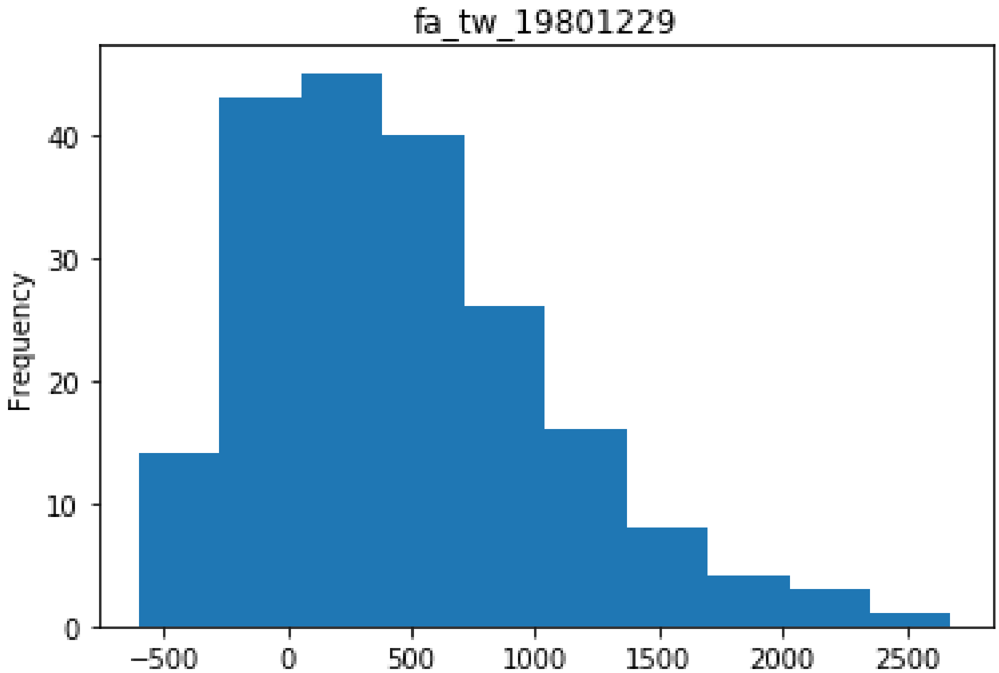
```
Out[8]: ['fa_hw_19791230',
         'fa_hw_19801229',
         'fa_tw_19791230',
         'fa_tw_19801229',
         'hds_00_013_002_000',
         'hds_00_013_002_001',
         'part_time',
         'part_status']
```

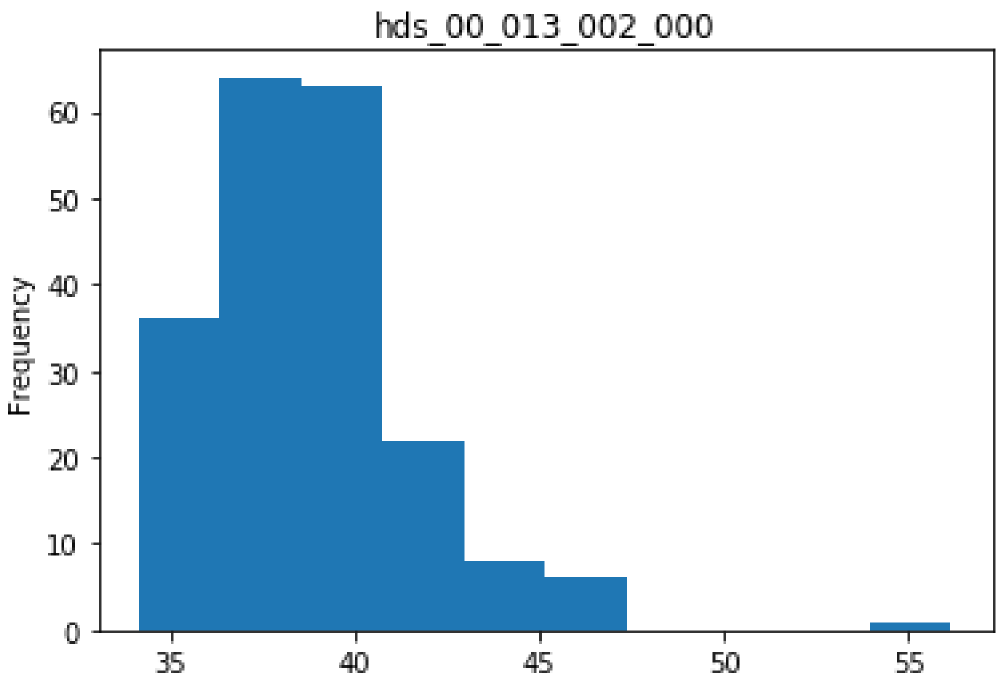
```
In [9]: for forecast in fnames:
            ax = obs_df.loc[:,forecast].plot(kind="hist")
            ax.set_title(forecast)
            plt.show()
```

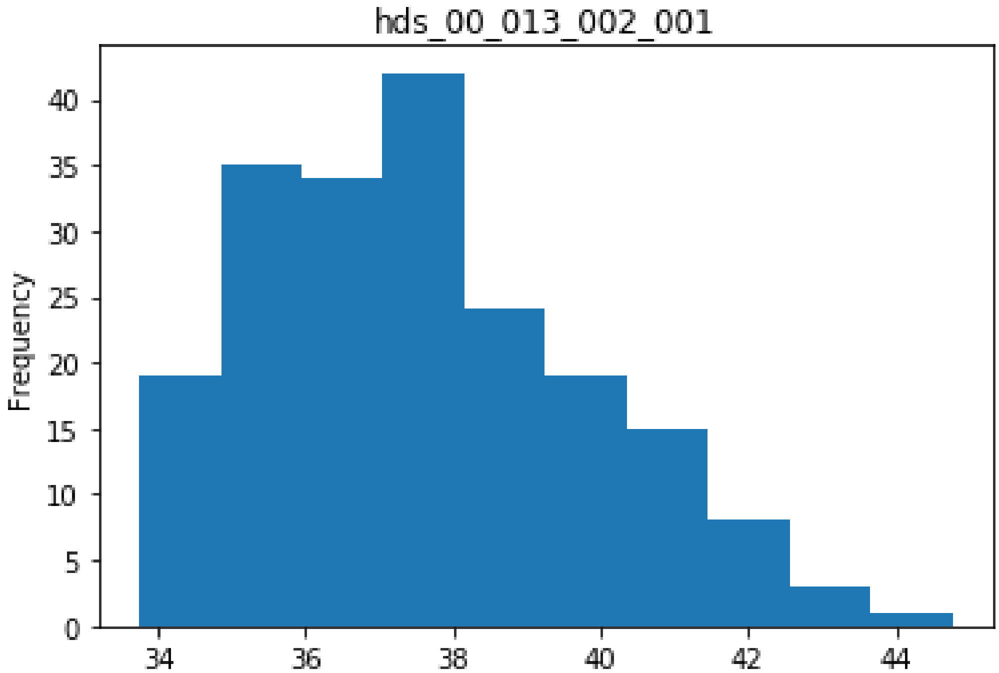


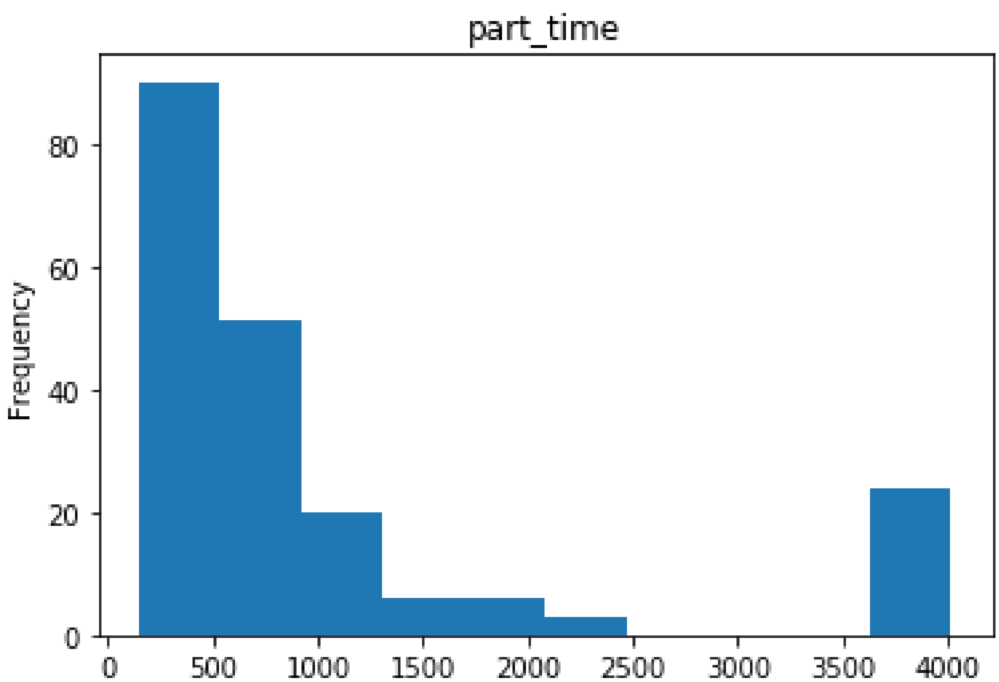

fa_hw_19801229
Frequency
0
10
20
30
40
50
60
70
−1000
−500
0
500
1000
1500
2000
2500

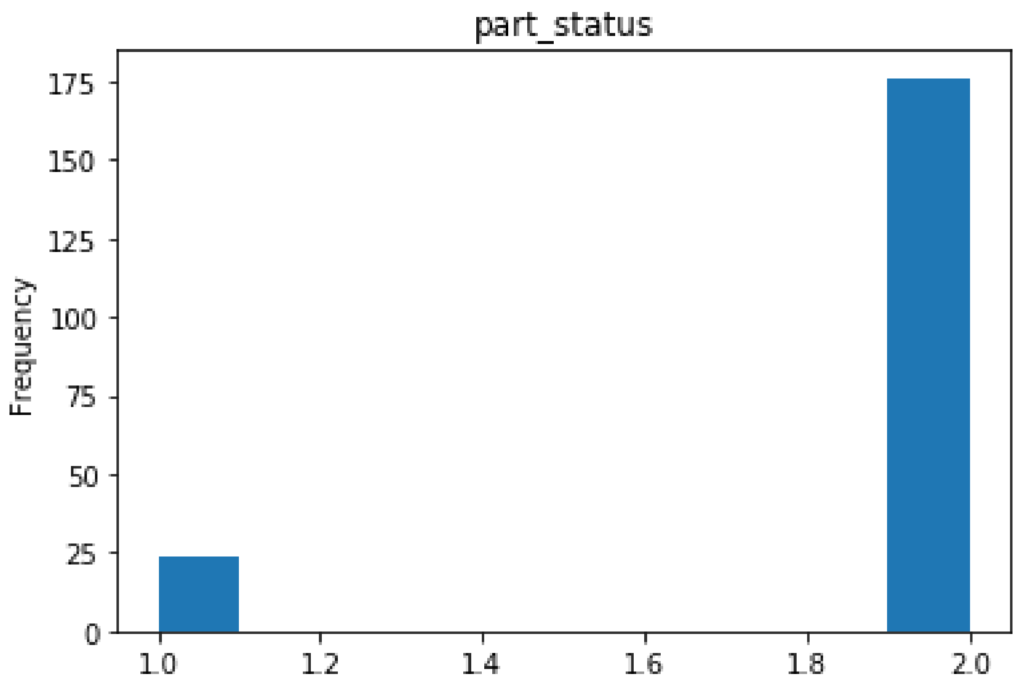

fa_tw_19791230
Frequency
0
10
20
30
40
50
−1000
−500
0
500
1000
1500


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


hds_00_013_002_000
Frequency
0
10
20
30
40
50
60
35
40
45
50
55

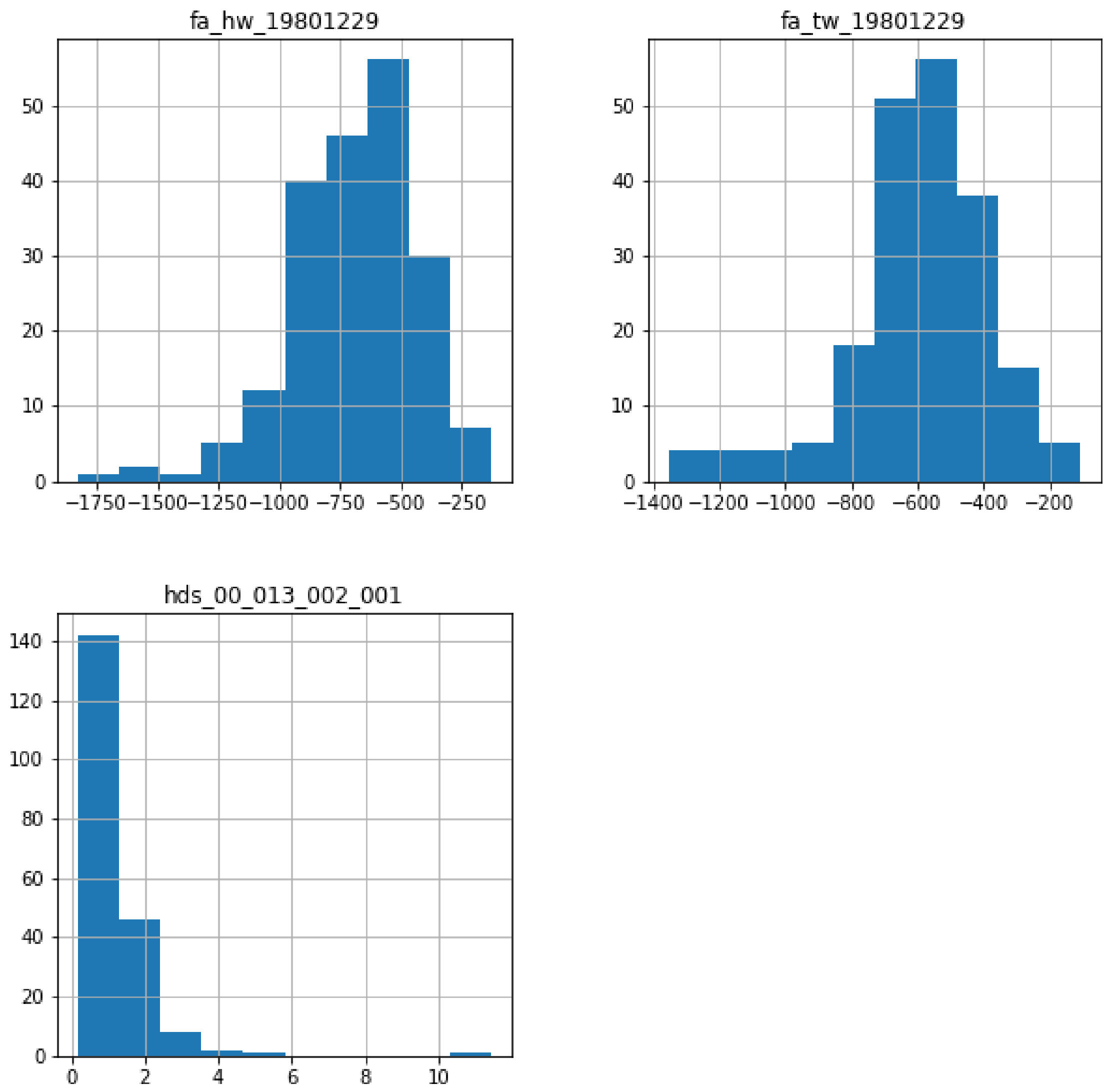

hds_00_013_002_001
Frequency
0
5
10
15
20
25
30
35
40
34
36
38
40
42
44


part_time
Frequency
0
20
40
60
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))

Out[10]: array([[<matplotlib.axes._subplots.AxesSubplot object at 0x1031c9a90>,
                 <matplotlib.axes._subplots.AxesSubplot object at 0x1031e9c88>],
                [<matplotlib.axes._subplots.AxesSubplot object at 0x1818cf0208>,
                 <matplotlib.axes._subplots.AxesSubplot object at 0x1818d06780>]],
               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[10]
idx
```

Out[11]: 23

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

```
Out[12]: fo_39_19791230          11553.000000
         hds_00_002_009_000         36.390945
         hds_00_002_015_000         34.901199
         hds_00_003_008_000         36.633041
         hds_00_009_001_000         38.894932
         hds_00_013_010_000         35.846367
         hds_00_015_016_000         34.909328
         hds_00_021_010_000         35.402809
         hds_00_022_015_000         34.446541
         hds_00_024_004_000         36.218941
         hds_00_026_006_000         35.067863
         hds_00_029_015_000         34.225922
         hds_00_033_007_000         34.120350
         hds_00_034_010_000         33.842617
         Name: 23, dtype: float64
```

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

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

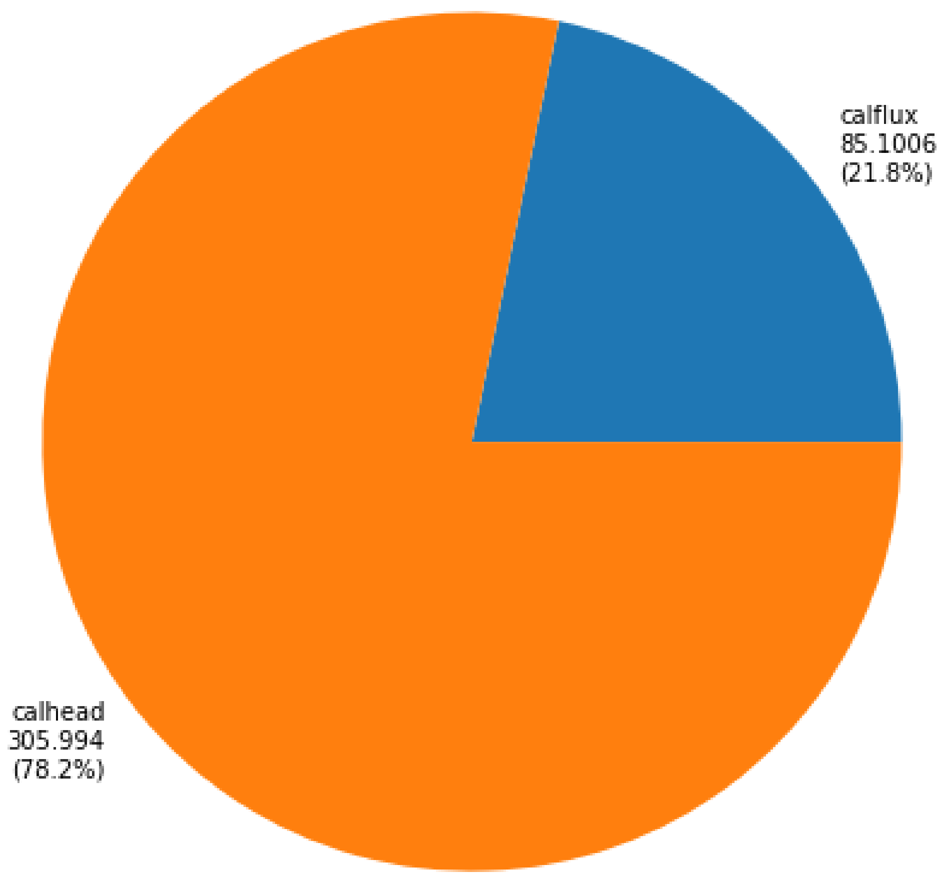
```
Out[13]: fa_hw_19791230        -1250.952650
         fa_hw_19801229         -383.912850
         fa_tw_19791230           97.419300
         fa_tw_19801229          841.068200
         hds_00_013_002_000       39.332623
         hds_00_013_002_001       37.060593
         part_time              1046.316000
         part_status               2.000000
         Name: 23, 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

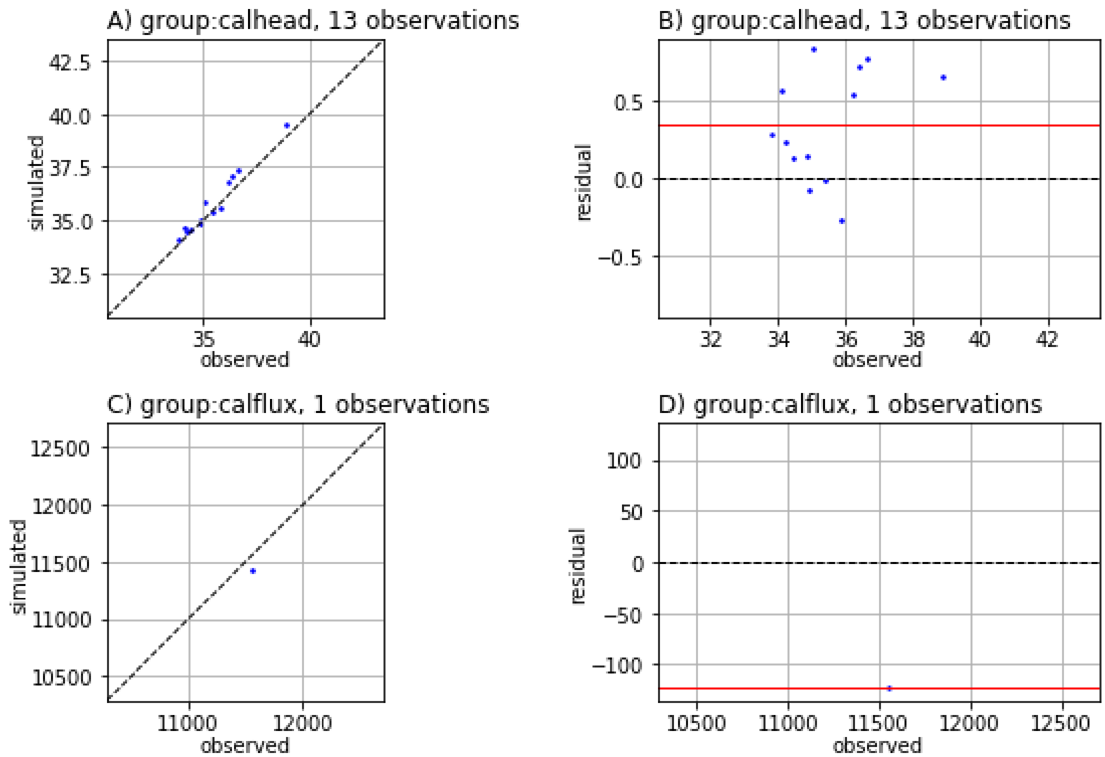
```
In [14]: pst = pyemu.Pst(os.path.join(t_d,"freyberg.pst"))
         obs = pst.observation_data
         obs.loc[:,"obsval"] = obs_df.loc[idx,pst.obs_names]
         obs.loc[obs.obgnme=="calhead","weight"] = 10.0
         obs.loc[obs.obgnme=="calflux","weight"] = 0.075
         np.random.seed(seed=0)
         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
         pst.write(os.path.join(t_d,"freyberg.pst"))
         pyemu.os_utils.run("pestpp-ies freyberg.pst",cwd=t_d)
```

```
In [15]: pst = pyemu.Pst(os.path.join(t_d,"freyberg.pst"))
         print(pst.phi)
         pst.plot(kind='phi_pie')
         pst.res.loc[pst.nnz_obs_names,:]
         figs = pst.plot(kind="1to1")
```

```
391.0941587400599
```




```
<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 [16]: 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,:]
```

```
8.972208849493734e-17
```

```
Out[16]:                                    name    group      measured      modelled  \
         name
         fo_39_19791230          fo_39_19791230  calflux  11553.000000  11553.000000
         hds_00_002_009_000  hds_00_002_009_000  calhead     36.390945     36.390945
         hds_00_002_015_000  hds_00_002_015_000  calhead     34.901199     34.901199
         hds_00_003_008_000  hds_00_003_008_000  calhead     36.633041     36.633041
         hds_00_009_001_000  hds_00_009_001_000  calhead     38.894932     38.894932
         hds_00_013_010_000  hds_00_013_010_000  calhead     35.846367     35.846367
         hds_00_015_016_000  hds_00_015_016_000  calhead     34.909328     34.909328
         hds_00_021_010_000  hds_00_021_010_000  calhead     35.402809     35.402809
         hds_00_022_015_000  hds_00_022_015_000  calhead     34.446541     34.446541
         hds_00_024_004_000  hds_00_024_004_000  calhead     36.218941     36.218941
         hds_00_026_006_000  hds_00_026_006_000  calhead     35.067863     35.067863
         hds_00_029_015_000  hds_00_029_015_000  calhead     34.225922     34.225922
         hds_00_033_007_000  hds_00_033_007_000  calhead     34.120350     34.120350
         hds_00_034_010_000  hds_00_034_010_000  calhead     33.842617     33.842617

                                 residual  weight
         name
         fo_39_19791230      0.000000e+00   0.075
         hds_00_002_009_000  4.296865e-10  10.000
         hds_00_002_015_000  1.796892e-10  10.000
         hds_00_003_008_000  1.640643e-10  10.000
         hds_00_009_001_000 -2.128928e-10  10.000
         hds_00_013_010_000 -3.242207e-10  10.000
         hds_00_015_016_000  3.066418e-10  10.000
         hds_00_021_010_000 -6.640732e-11  10.000
         hds_00_022_015_000  4.804690e-10  10.000
         hds_00_024_004_000  1.367155e-10  10.000
         hds_00_026_006_000 -3.554703e-10  10.000
         hds_00_029_015_000  1.406235e-10  10.000
         hds_00_033_007_000 -3.320366e-11  10.000
         hds_00_034_010_000  8.789414e-11  10.000
```

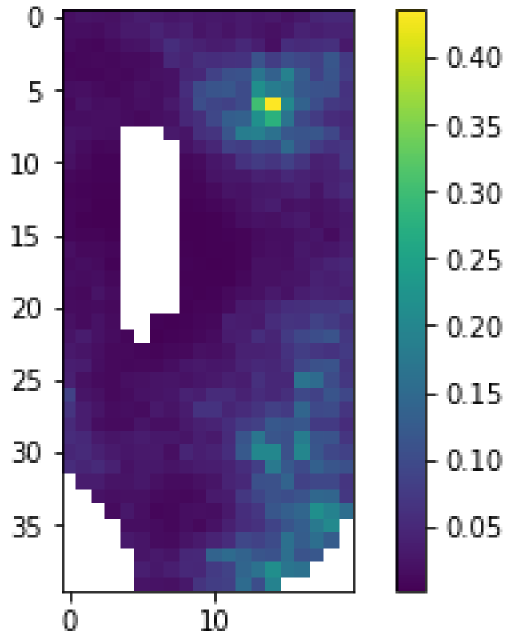
The residual should be exactly the noise values from above.Lets load the model (that was just run using the true pars) and check some things

```
In [17]: m = flopy.modflow.Modflow.load("freyberg.nam",model_ws=m_d)
```

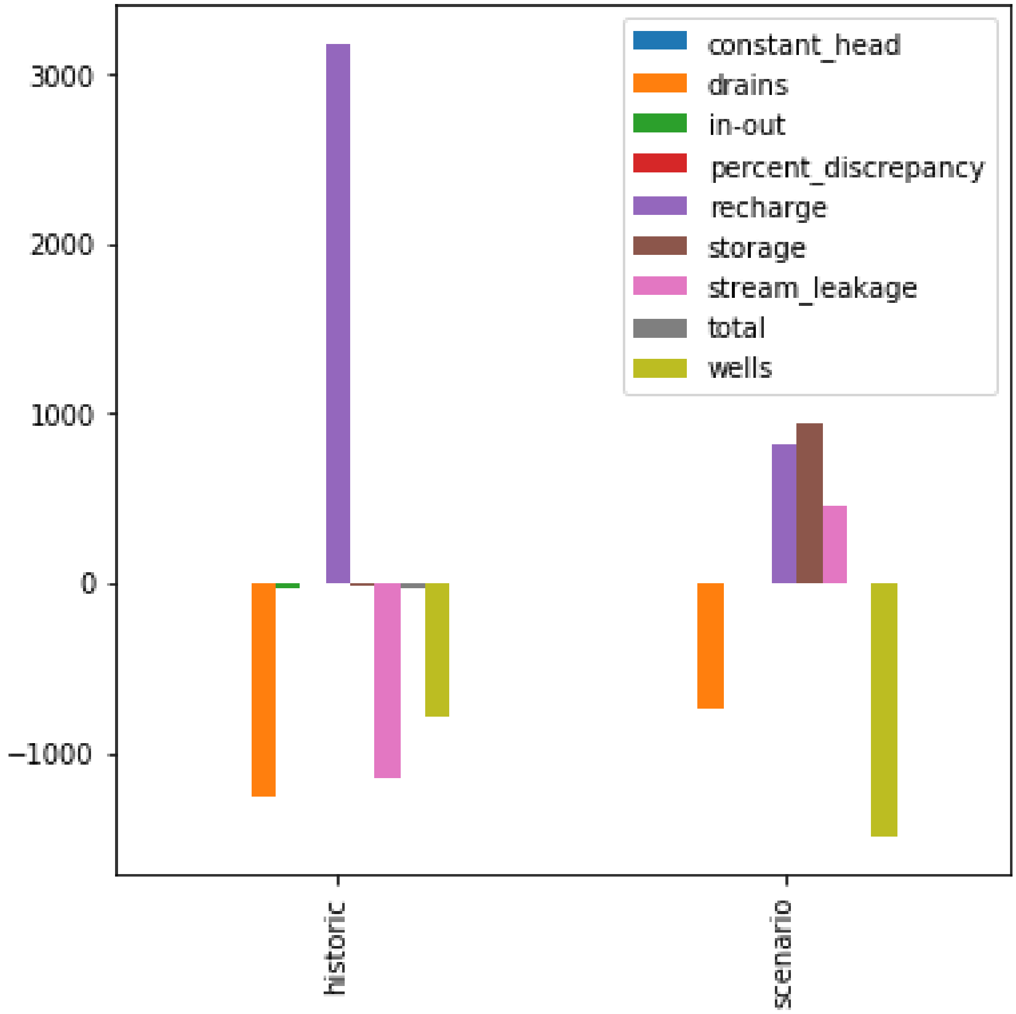
In [18]:
```
a = m.upw.vka[1].array
#a = m.rch.rech[0].array
a = np.ma.masked_where(m.bas6.ibound[0].array==0,a)
print(a.min(),a.max())
c = plt.imshow(a)
plt.colorbar(c)
```

```
0.002020436 0.4354167
```

Out[18]:
```
<matplotlib.colorbar.Colorbar at 0x1819bcd208>
```



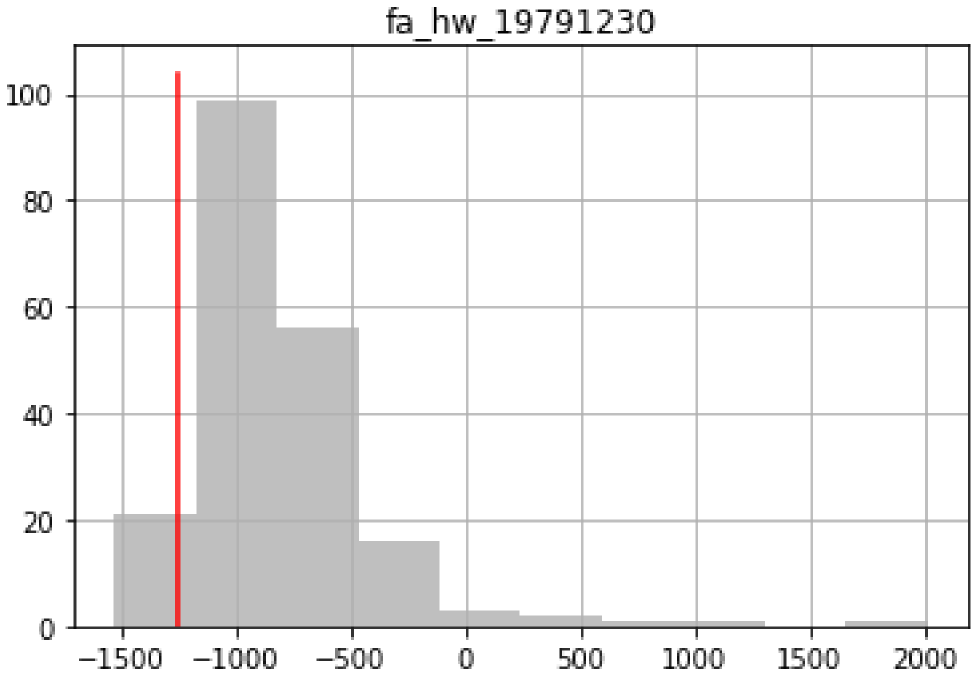
In [19]:
```
lst = flopy.utils.MfListBudget(os.path.join(m_d,"freyberg.list"))
df = lst.get_dataframes(diff=True)[0]
ax = df.plot(kind="bar",figsize=(6,6))
a = ax.set_xticklabels(["historic","scenario"],rotation=90)
```

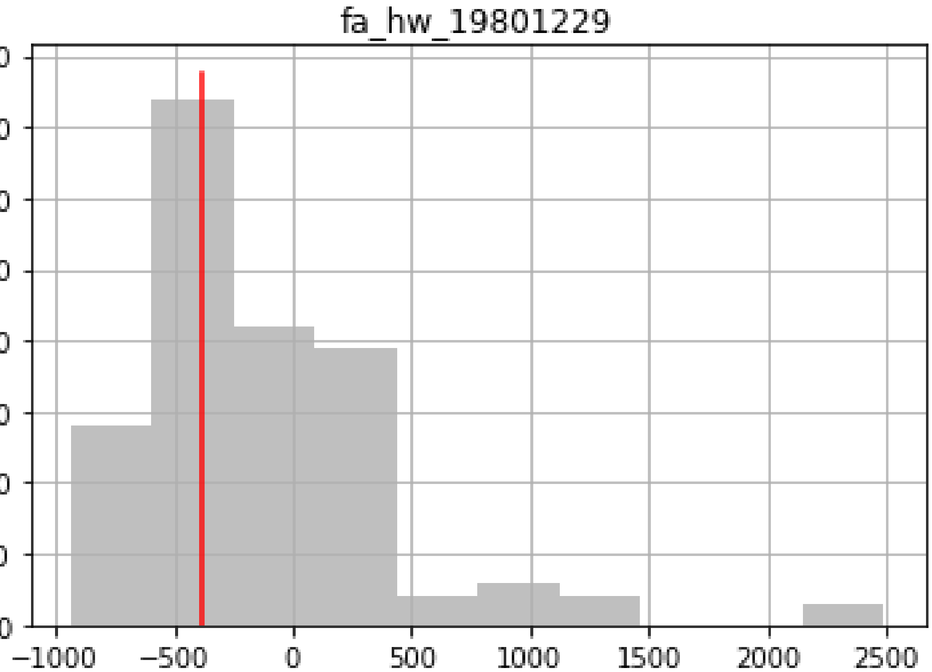


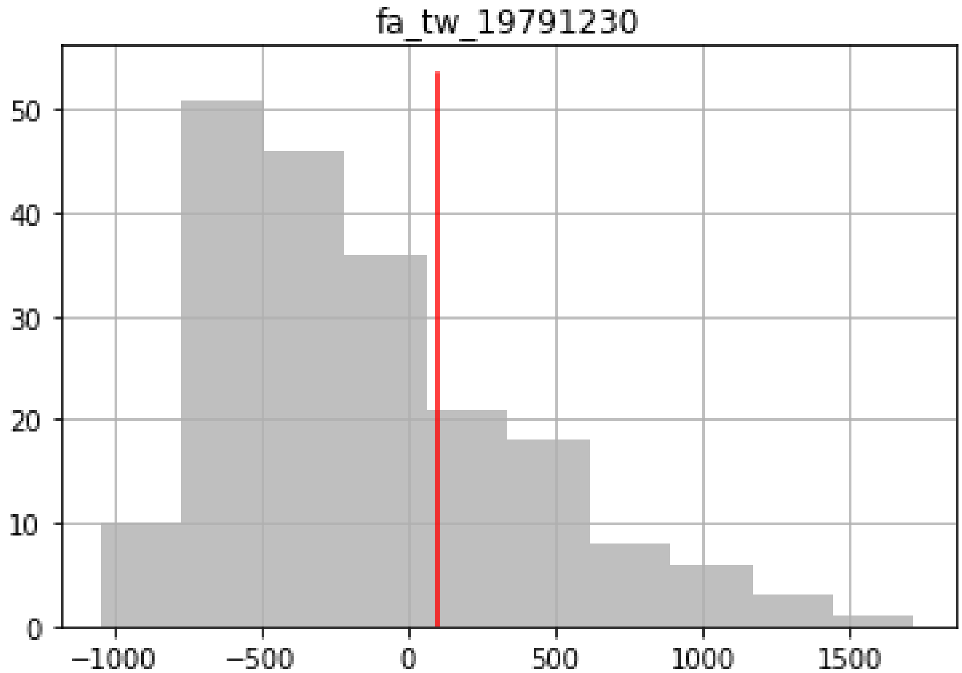
1.0.5 see how our existing observation ensemble compares to the truth

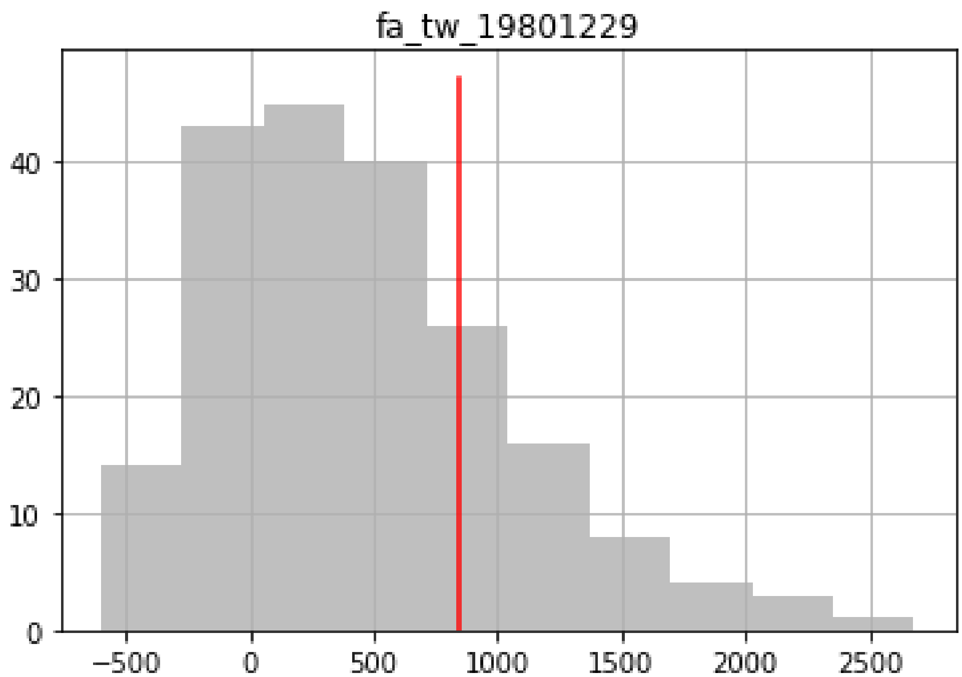
sw-gw outputs:

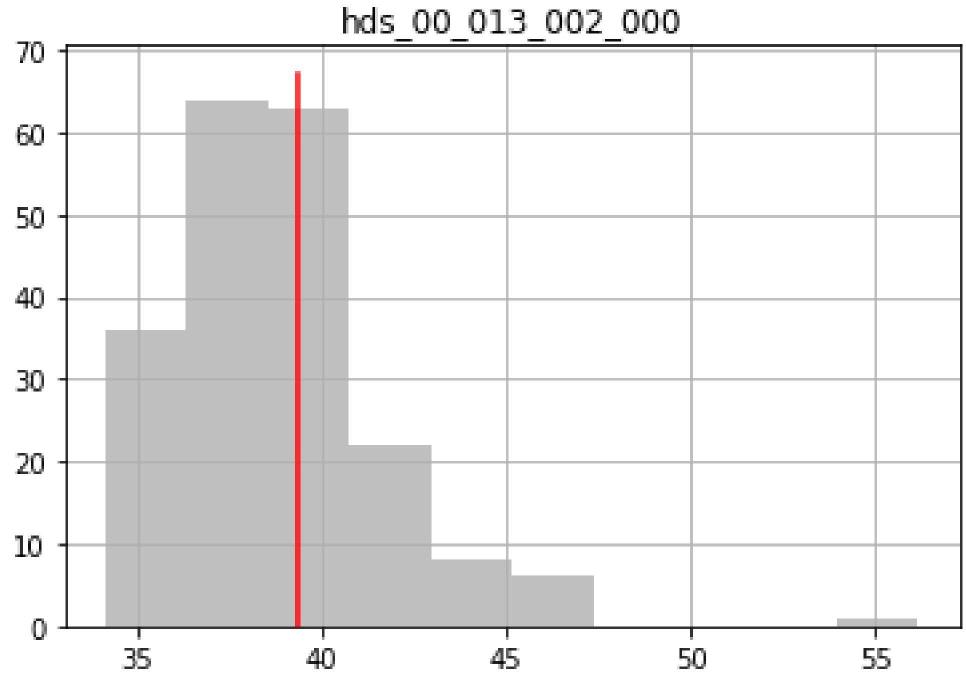
```
In [20]: obs = pst.observation_data

         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()
```

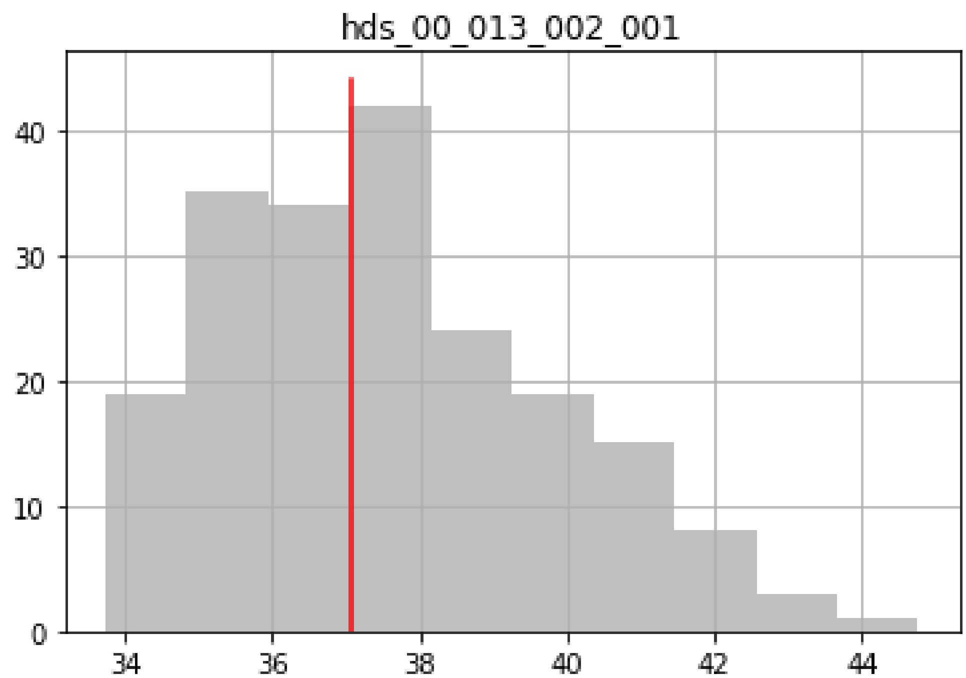

fa_hw_19791230
100
80
60
40
20
0
−1500
−1000
−500
0
500
1000
1500
2000

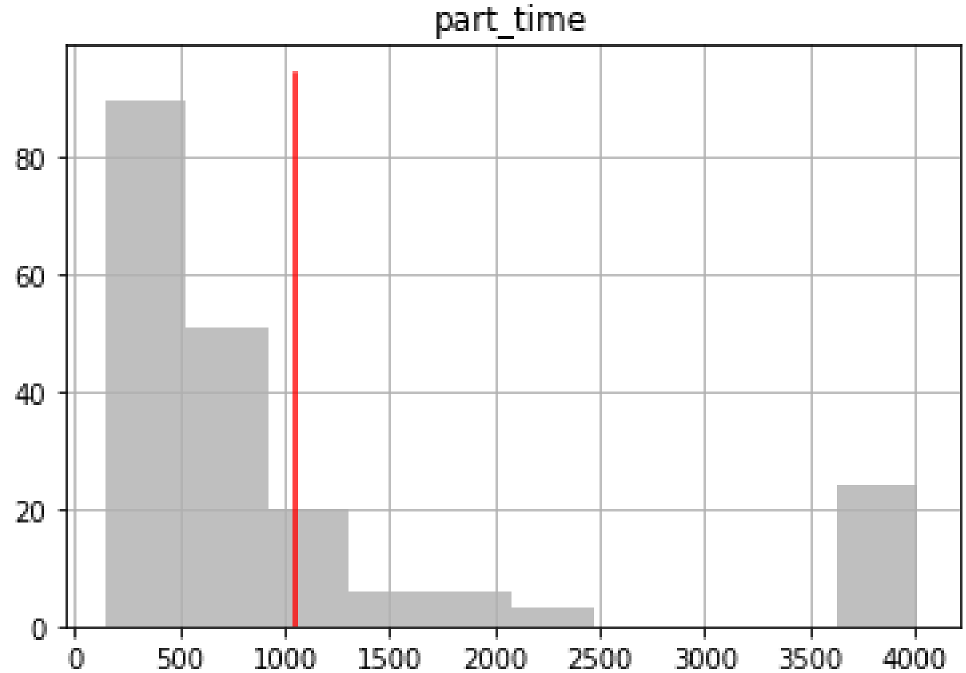

fa_hw_19801229
80
70
60
50
40
30
20
10
0
−1000
−500
0
500
1000
1500
2000
2500

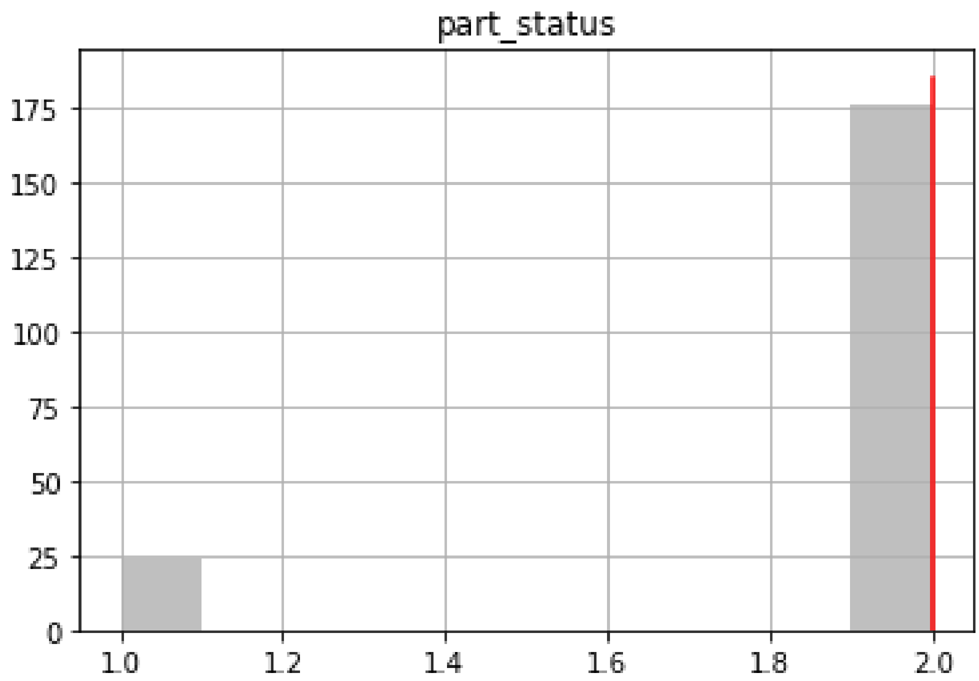

fa_tw_19791230
50
40
30
20
10
0
−1000
−500
0
500
1000
1500


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


hds_00_013_002_000
70
60
50
40
30
20
10
0
35
40
45
50
55

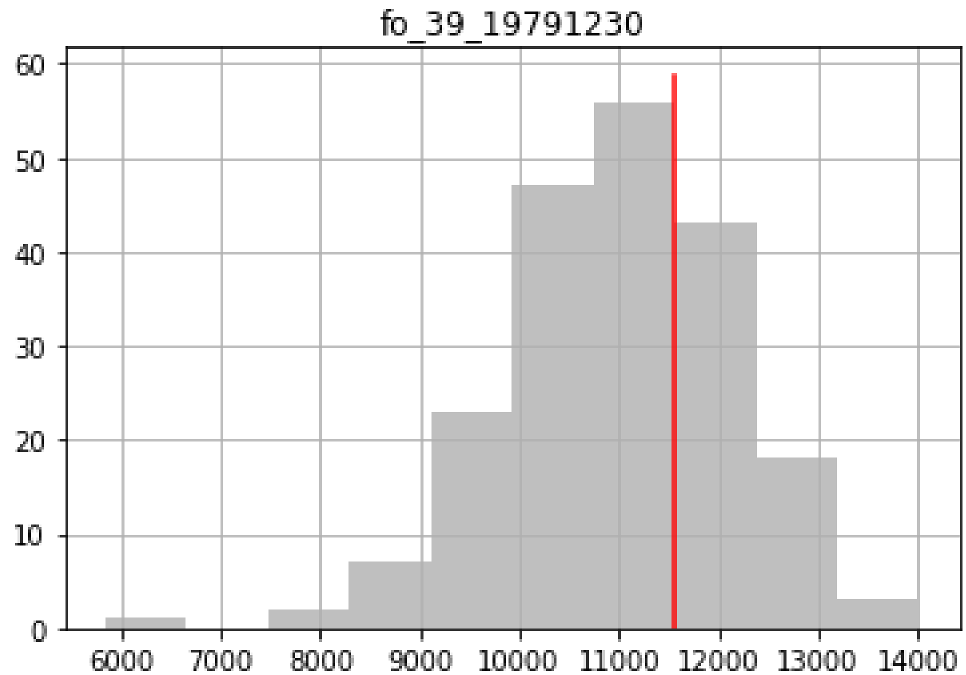

hds_00_013_002_001
40
30
20
10
0
34
36
38
40
42
44

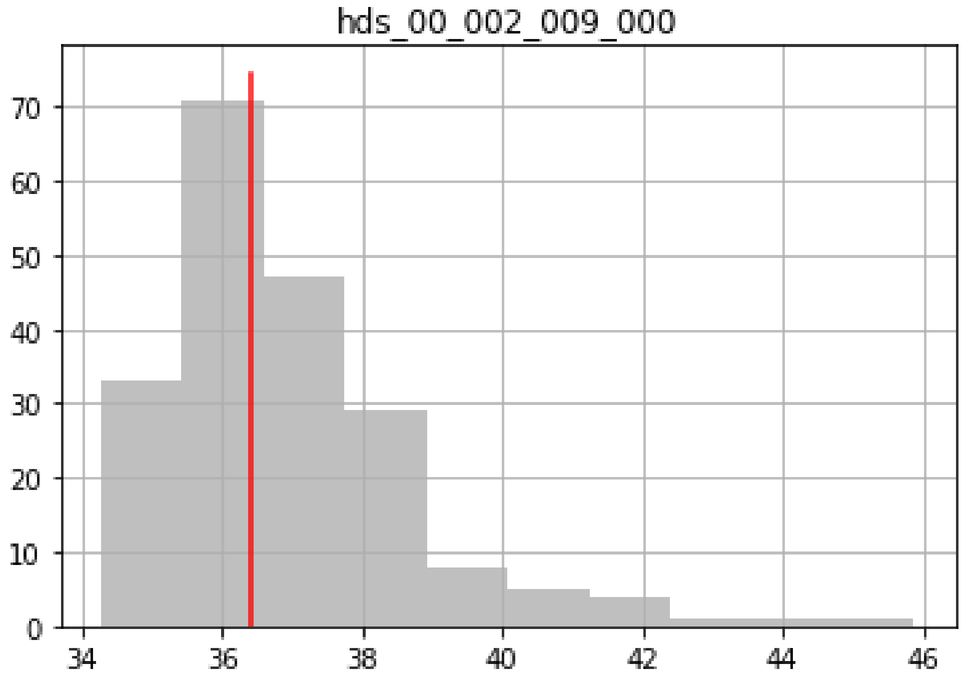


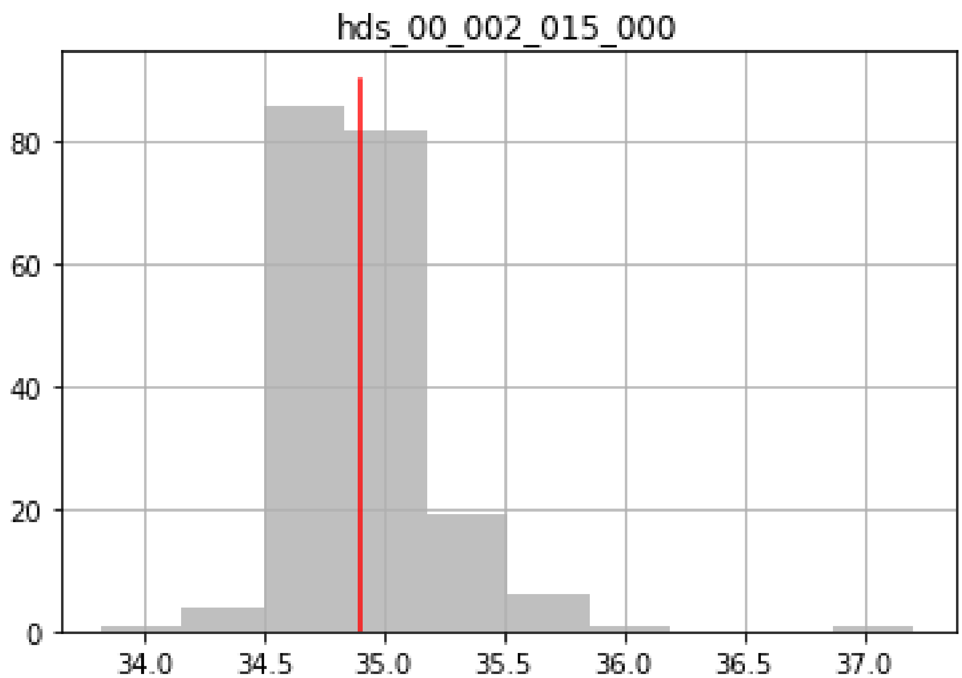


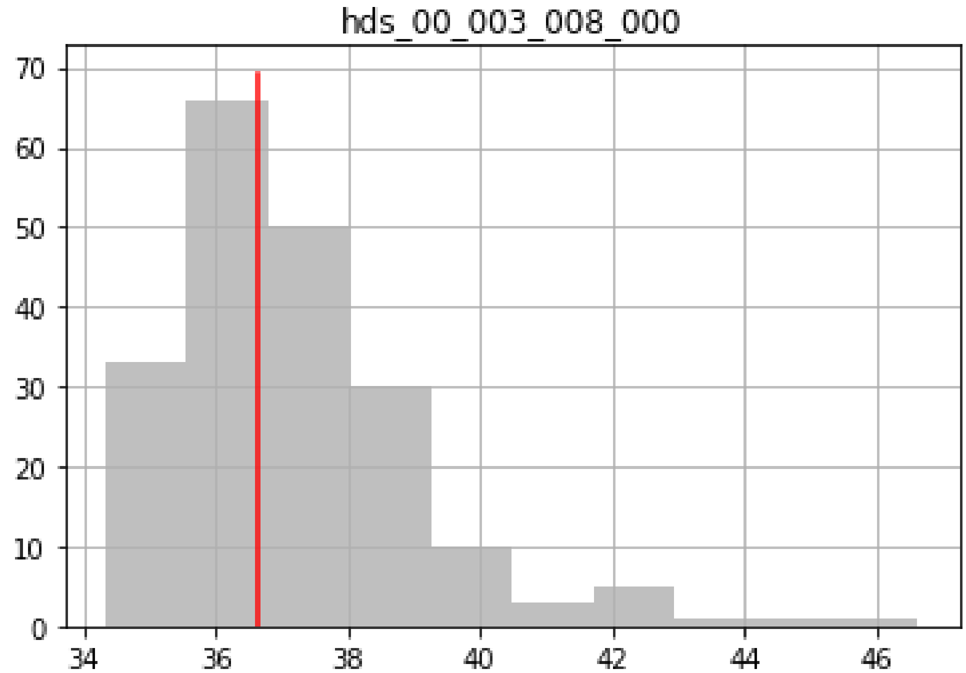
observations:

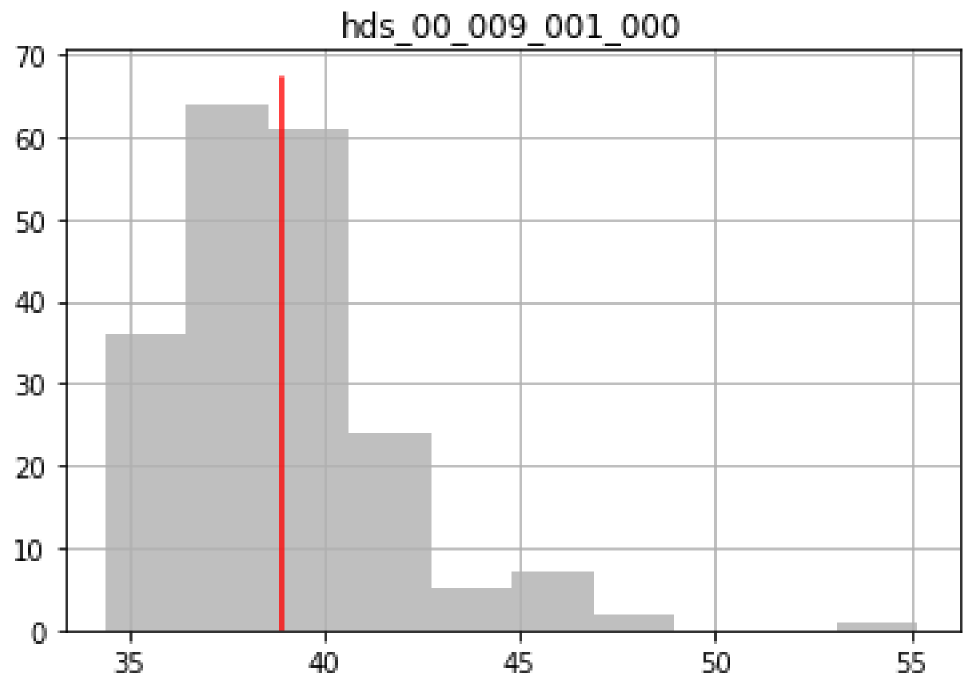
In [21]:
```
for oname in pst.nnz_obs_names:
    ax = plt.subplot(111)
    obs_df.loc[:,oname].hist(ax=ax,color="0.5",alpha=0.5)
    ax.plot([obs.loc[oname,"obsval"],obs.loc[oname,"obsval"]],ax.get_ylim(),"r")
    ax.set_title(oname)
    plt.show()
```

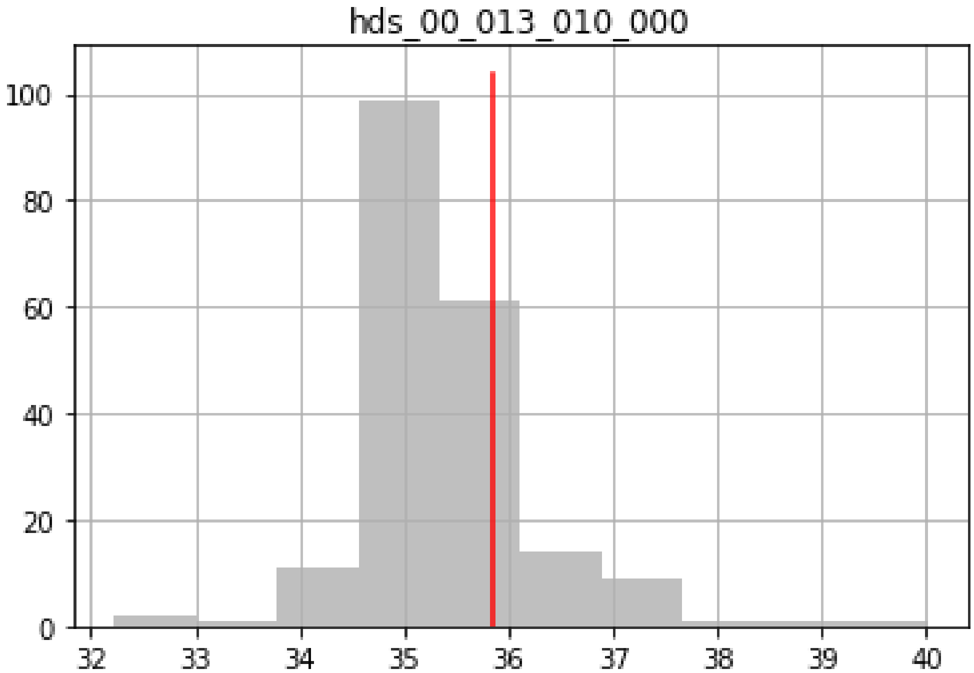


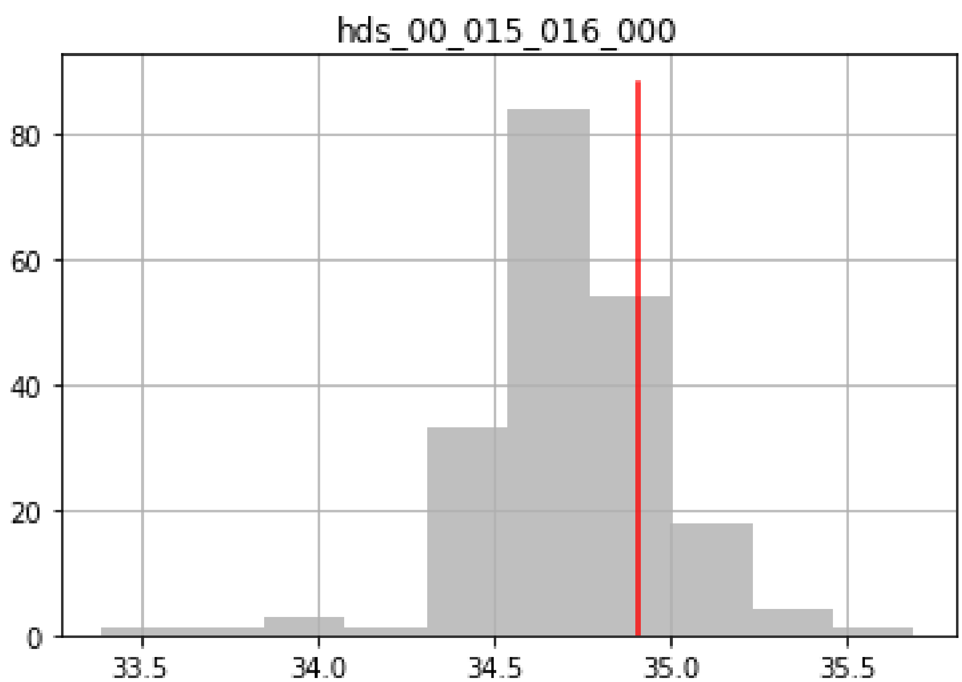

hds_00_002_009_000
70
60
50
40
30
20
10
0
34
36
38
40
42
44
46

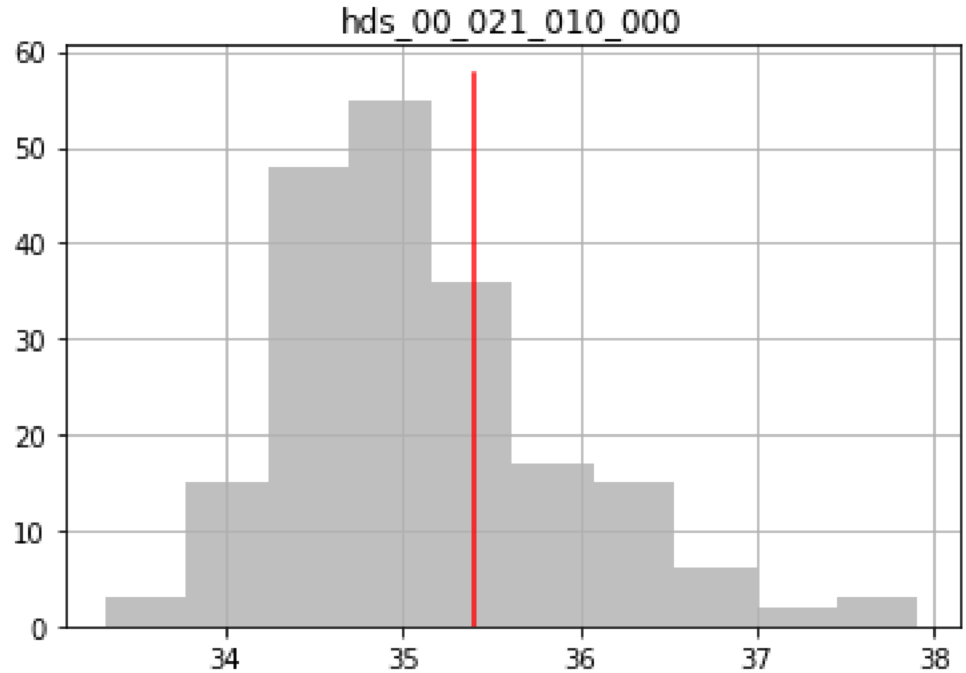

hds_00_002_015_000
80
60
40
20
0
34.0
34.5
35.0
35.5
36.0
36.5
37.0

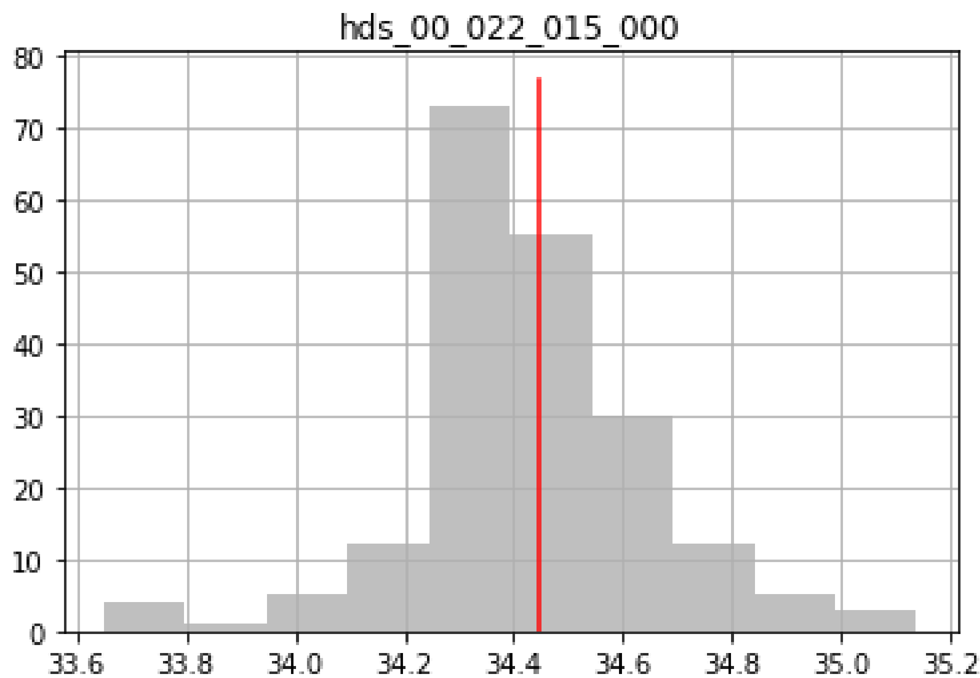

hds_00_003_008_000
0
10
20
30
40
50
60
70
34
36
38
40
42
44
46

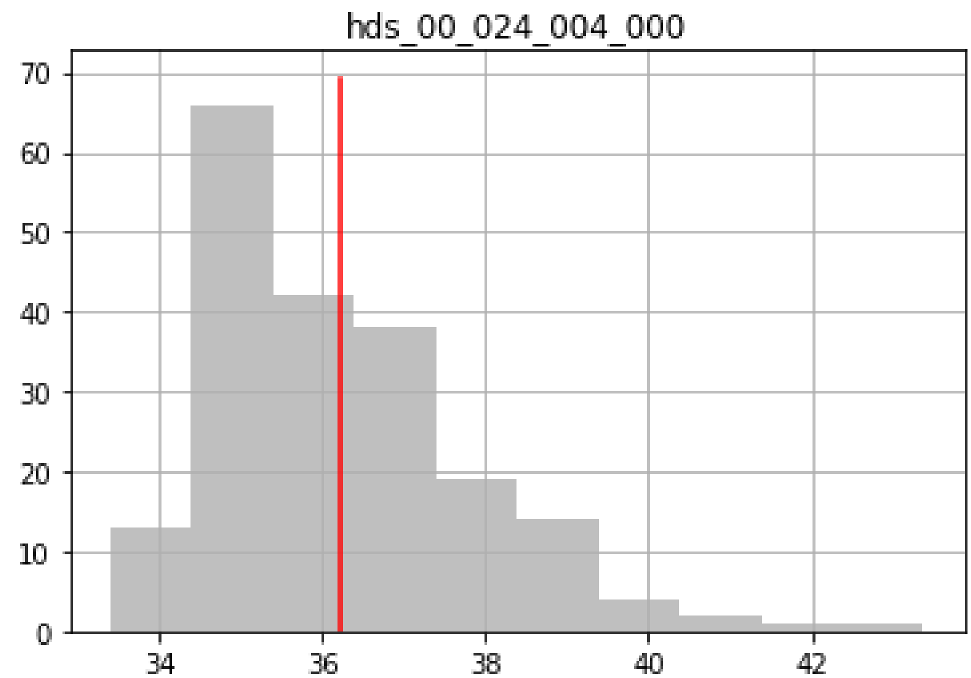

hds_00_009_001_000
0
10
20
30
40
50
60
70
35
40
45
50
55

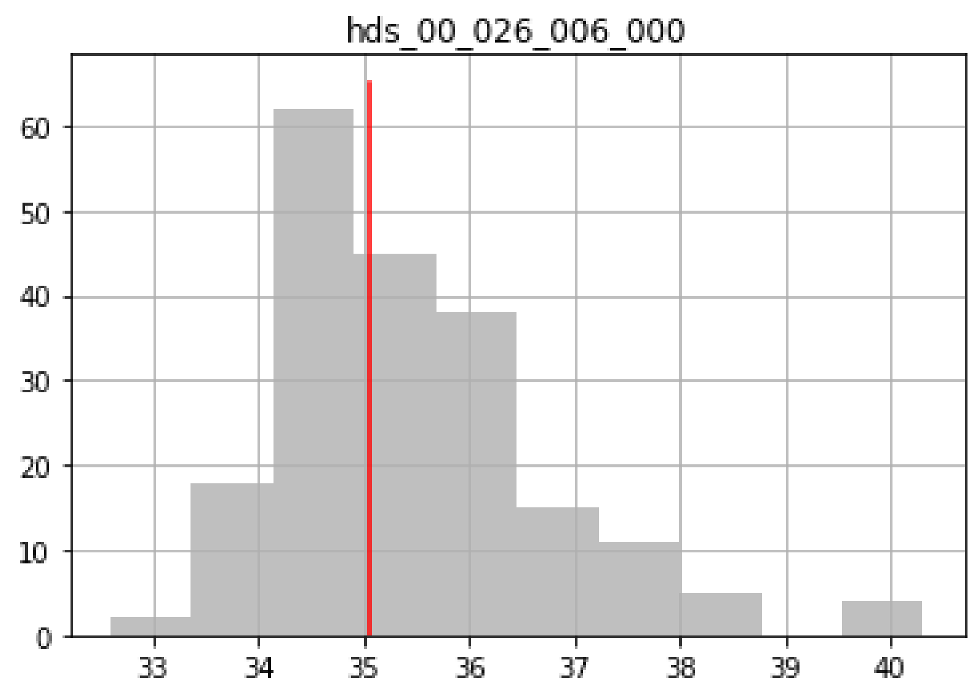

hds_00_013_010_000
100
80
60
40
20
0
32
33
34
35
36
37
38
39
40

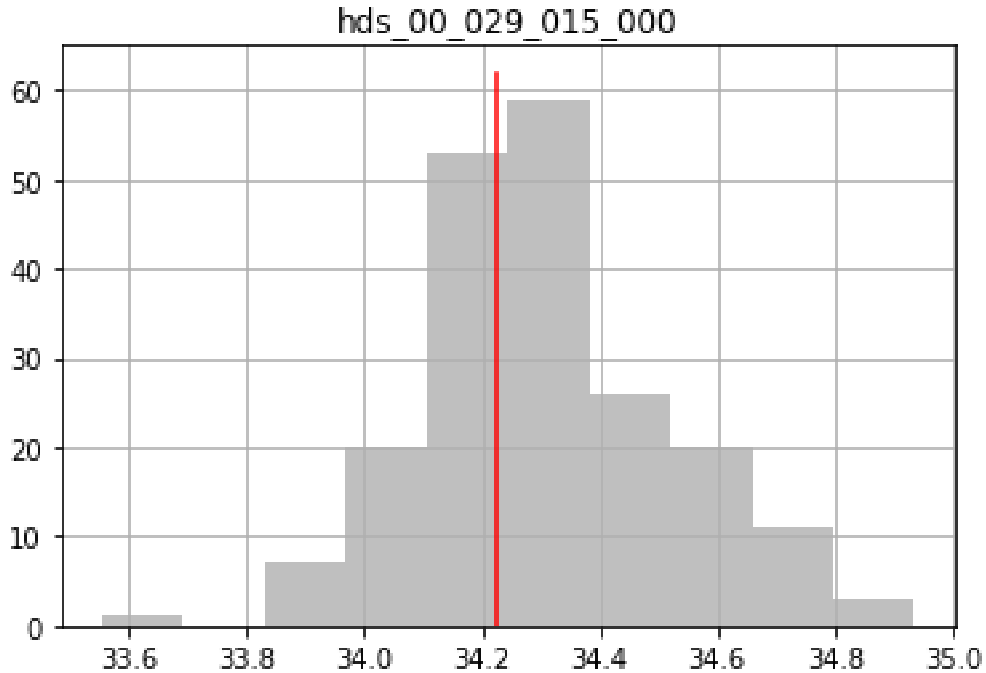

hds_00_015_016_000
80
60
40
20
0
33.5
34.0
34.5
35.0
35.5

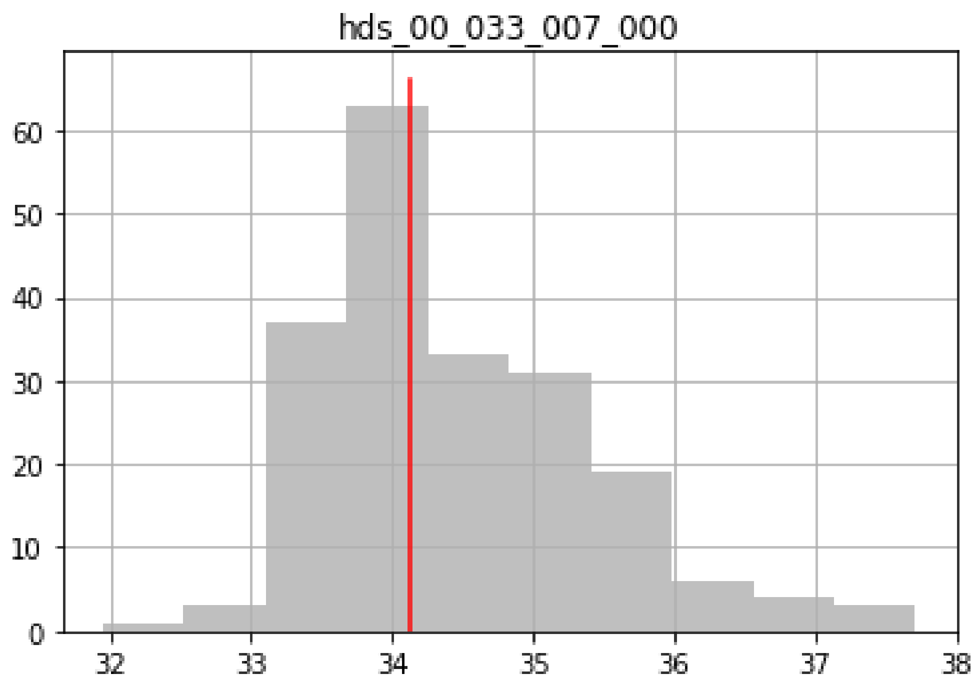

hds_00_021_010_000
0
10
20
30
40
50
60
34
35
36
37
38

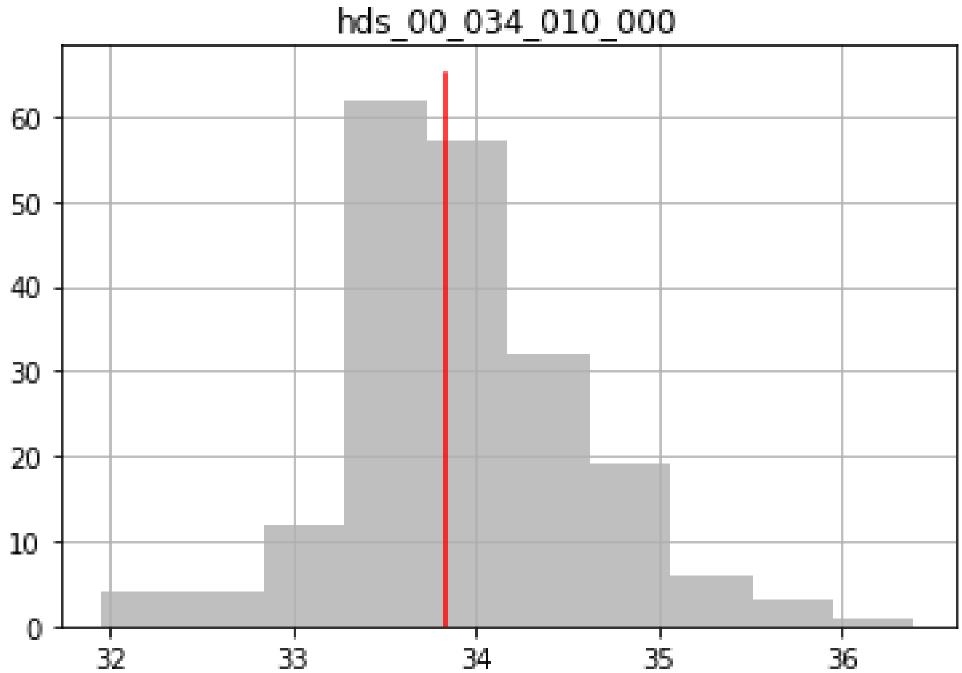

hds_00_022_015_000
0
10
20
30
40
50
60
70
80
33.6
33.8
34.0
34.2
34.4
34.6
34.8
35.0
35.2


hds_00_024_004_000
70
60
50
40
30
20
10
0
34
36
38
40
42


hds_00_026_006_000
60
50
40
30
20
10
0
33
34
35
36
37
38
39
40


hds_00_029_015_000
0
10
20
30
40
50
60
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
32
33
34
35
36
37
38


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