prior_montecarlo

May 14, 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...
```

In [5]:
```
pe.loc[:,"hk031"]
```

Out[5]:

0	1.825601
1	0.492395
2	1.732541
3	0.705128
4	2.297188
5	1.252259
6	0.516871
7	2.804304
8	0.901501
9	0.687264
10	0.560744
11	1.109859
12	3.769390
13	5.603115
14	0.498500
15	1.641592
16	1.671183
17	0.789444
18	2.699144
19	0.597935
20	0.509815
21	0.599028
22	1.237666
23	1.282956
24	0.626140
25	1.804318
26	0.800346
27	3.591938
28	1.315121
29	0.407926
...	...
470	2.182434
471	0.428269
472	0.315921
473	1.160333
474	2.349660
475	4.092183
476	0.965726
477	0.541814

```
478    0.353042
479    0.988625
480    0.292822
481    0.369399
482    2.130098
483    0.641830
484    1.972197
485    0.783833
486    2.121167
487    4.772154
488    1.429077
489    2.661986
490    0.995772
491    1.386897
492    1.510872
493    1.115474
494    2.634503
495    1.353126
496    1.003978
497    0.668727
498    0.492995
499    1.109220
Name: hk031, Length: 500, dtype: float64
```

```
In [6]: pst.parameter_data.loc[pe.columns,"parval1"] = pe.iloc[0,:]
        pst.control_data.noptmax = 0
        pst.write(os.path.join(t_d,"test.pst"))
        pyemu.os_utils.run("pestpp-ies test.pst",cwd=t_d)
        res = pyemu.pst_utils.read_resfile(os.path.join(t_d,"test.base.rei"))
        res
```

```
noptmax:0, npar_adj:14819, nnz_obs:14
```

Out[6]:

name	name	group	measured	\
fa_0_19791230	fa_0_19791230	flaqx	-6.907900e+01	
fa_0_19801229	fa_0_19801229	flaqx	-6.895800e+01	
fa_10_19791230	fa_10_19791230	flaqx	-3.626600e+01	
fa_10_19801229	fa_10_19801229	flaqx	-3.620300e+01	
fa_11_19791230	fa_11_19791230	flaqx	-3.737100e+01	
fa_11_19801229	fa_11_19801229	flaqx	-3.731600e+01	
fa_12_19791230	fa_12_19791230	flaqx	-4.045900e+01	
fa_12_19801229	fa_12_19801229	flaqx	-4.041100e+01	
fa_13_19791230	fa_13_19791230	flaqx	-4.308200e+01	
fa_13_19801229	fa_13_19801229	flaqx	-4.303900e+01	
fa_14_19791230	fa_14_19791230	flaqx	-4.471700e+01	
fa_14_19801229	fa_14_19801229	flaqx	-4.467800e+01	

```
fa_15_19791230              fa_15_19791230        flaqx -4.523300e+01
fa_15_19801229              fa_15_19801229        flaqx -4.519800e+01
fa_16_19791230              fa_16_19791230        flaqx -4.498900e+01
fa_16_19801229              fa_16_19801229        flaqx -4.495700e+01
fa_17_19791230              fa_17_19791230        flaqx -4.367400e+01
fa_17_19801229              fa_17_19801229        flaqx -4.364200e+01
fa_18_19791230              fa_18_19791230        flaqx -4.095300e+01
fa_18_19801229              fa_18_19801229        flaqx -4.092200e+01
fa_19_19791230              fa_19_19791230        flaqx -3.618200e+01
fa_19_19801229              fa_19_19801229        flaqx -3.615100e+01
fa_1_19791230                fa_1_19791230        flaqx -6.944200e+01
fa_1_19801229                fa_1_19801229        flaqx -6.932200e+01
fa_20_19791230              fa_20_19791230        flaqx -3.008600e+01
fa_20_19801229              fa_20_19801229        flaqx -3.005500e+01
fa_21_19791230              fa_21_19791230        flaqx -3.548400e+01
fa_21_19801229              fa_21_19801229        flaqx -3.545200e+01
fa_22_19791230              fa_22_19791230        flaqx -3.935200e+01
fa_22_19801229              fa_22_19801229        flaqx -3.931800e+01
...                                    ...          ...           ...
hds_02_039_010_000      hds_02_039_010_000          hds  3.256046e+01
hds_02_039_010_001      hds_02_039_010_001          hds  3.256043e+01
hds_02_039_011_000      hds_02_039_011_000          hds  3.256142e+01
hds_02_039_011_001      hds_02_039_011_001          hds  3.256139e+01
hds_02_039_012_000      hds_02_039_012_000          hds  3.256558e+01
hds_02_039_012_001      hds_02_039_012_001          hds  3.256556e+01
hds_02_039_013_000      hds_02_039_013_000          hds  3.257711e+01
hds_02_039_013_001      hds_02_039_013_001          hds  3.257710e+01
hds_02_039_014_000      hds_02_039_014_000          hds  3.259781e+01
hds_02_039_014_001      hds_02_039_014_001          hds  3.259779e+01
vol_constan_19791230  vol_constan_19791230  vol_constan  0.000000e+00
vol_constan_19801229  vol_constan_19801229  vol_constan  0.000000e+00
vol_drains_19791230    vol_drains_19791230   vol_drains -2.640137e+06
vol_drains_19801229    vol_drains_19801229   vol_drains -2.904042e+06
vol_in-out_19791230    vol_in-out_19791230   vol_in-out  4.500000e+01
vol_in-out_19801229    vol_in-out_19801229   vol_in-out  6.300000e+01
vol_percent_19791230  vol_percent_19791230  vol_percent  0.000000e+00
vol_percent_19801229  vol_percent_19801229  vol_percent  0.000000e+00
vol_recharg_19791230  vol_recharg_19791230  vol_recharg  1.111644e+07
vol_recharg_19801229  vol_recharg_19801229  vol_recharg  1.222808e+07
vol_storage_19791230  vol_storage_19791230  vol_storage  2.923828e+04
vol_storage_19801229  vol_storage_19801229  vol_storage  3.134556e+04
vol_stream__19791230  vol_stream__19791230  vol_stream_ -5.220494e+06
vol_stream__19801229  vol_stream__19801229  vol_stream_ -5.741824e+06
vol_total_19791230      vol_total_19791230    vol_total  4.500000e+01
vol_total_19801229      vol_total_19801229    vol_total  6.300000e+01
vol_wells_19791230      vol_wells_19791230    vol_wells -3.285000e+06
vol_wells_19801229      vol_wells_19801229    vol_wells -3.613500e+06
part_status                    part_status       obgnme  1.000000e+10
```

part_time part_time obgnme 1.000000e+10

name	modelled	residual	weight
fa_0_19791230	-8.373800e+01	1.465900e+01	0.0
fa_0_19801229	-3.831900e+01	-3.063900e+01	0.0
fa_10_19791230	5.866700e+00	-4.213270e+01	0.0
fa_10_19801229	5.919200e+01	-9.539500e+01	0.0
fa_11_19791230	-1.928500e+01	-1.808600e+01	0.0
fa_11_19801229	6.741300e+01	-1.047290e+02	0.0
fa_12_19791230	-4.867200e+01	8.213000e+00	0.0
fa_12_19801229	1.706000e+01	-5.747100e+01	0.0
fa_13_19791230	-3.496600e+01	-8.116000e+00	0.0
fa_13_19801229	-2.524200e+00	-4.051480e+01	0.0
fa_14_19791230	-5.240200e+01	7.685000e+00	0.0
fa_14_19801229	-1.058200e+01	-3.409600e+01	0.0
fa_15_19791230	-3.013900e+01	-1.509400e+01	0.0
fa_15_19801229	-7.566100e+00	-3.763190e+01	0.0
fa_16_19791230	-3.813200e+01	-6.857000e+00	0.0
fa_16_19801229	-1.062900e+01	-3.432800e+01	0.0
fa_17_19791230	-9.060900e+01	4.693500e+01	0.0
fa_17_19801229	-2.660900e+01	-1.703300e+01	0.0
fa_18_19791230	-3.301300e+01	-7.940000e+00	0.0
fa_18_19801229	-9.515400e+00	-3.140660e+01	0.0
fa_19_19791230	-6.556500e+01	2.938300e+01	0.0
fa_19_19801229	-1.126000e+01	-2.489100e+01	0.0
fa_1_19791230	-1.725800e+01	-5.218400e+01	0.0
fa_1_19801229	-8.194200e+00	-6.112780e+01	0.0
fa_20_19791230	-2.728800e+01	-2.798000e+00	0.0
fa_20_19801229	1.592500e+01	-4.598000e+01	0.0
fa_21_19791230	-4.538500e+01	9.901000e+00	0.0
fa_21_19801229	-9.436800e+00	-2.601520e+01	0.0
fa_22_19791230	-2.906400e+01	-1.028800e+01	0.0
fa_22_19801229	-1.059700e+01	-2.872100e+01	0.0
...	...	...	...
hds_02_039_010_000	3.258691e+01	-2.645493e-02	0.0
hds_02_039_010_001	3.254329e+01	1.713943e-02	0.0
hds_02_039_011_000	3.256671e+01	-5.287170e-03	0.0
hds_02_039_011_001	3.253341e+01	2.798462e-02	0.0
hds_02_039_012_000	3.255355e+01	1.203155e-02	0.0
hds_02_039_012_001	3.252919e+01	3.636932e-02	0.0
hds_02_039_013_000	3.255584e+01	2.127075e-02	0.0
hds_02_039_013_001	3.253627e+01	4.082489e-02	0.0
hds_02_039_014_000	3.256308e+01	3.472519e-02	0.0
hds_02_039_014_001	3.254853e+01	4.925537e-02	0.0
vol_constan_19791230	0.000000e+00	0.000000e+00	0.0
vol_constan_19801229	0.000000e+00	0.000000e+00	0.0
vol_drains_19791230	-2.426614e+06	-2.135235e+05	0.0

```
vol_drains_19801229  -2.548770e+06 -3.552720e+05     0.0
vol_in-out_19791230  -2.196660e+05  2.197110e+05     0.0
vol_in-out_19801229  -2.192100e+05  2.192730e+05     0.0
vol_percent_19791230 -1.780000e+00  1.780000e+00     0.0
vol_percent_19801229 -1.670000e+00  1.670000e+00     0.0
vol_recharg_19791230  1.164774e+07 -5.312960e+05     0.0
vol_recharg_19801229  1.190907e+07  3.190110e+05     0.0
vol_storage_19791230  4.638466e+05 -4.346083e+05     0.0
vol_storage_19801229  9.305074e+05 -8.991619e+05     0.0
vol_stream__19791230 -5.401493e+06  1.809990e+05     0.0
vol_stream__19801229 -5.440004e+06 -3.018195e+05     0.0
vol_total_19791230   -2.196660e+05  2.197110e+05     0.0
vol_total_19801229   -2.192100e+05  2.192730e+05     0.0
vol_wells_19791230   -4.503142e+06  1.218142e+06     0.0
vol_wells_19801229   -5.070016e+06  1.456516e+06     0.0
part_status           2.000000e+00  1.000000e+10     0.0
part_time             1.988810e+03  9.999998e+09     0.0

[4436 rows x 6 columns]
```

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 [7]: 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 [8]: 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: 500
```

drop any failed runs

```
In [9]: 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: 500
```

```
In [10]: obs_df.iloc[0,:]
```

```
Out[10]: input_run_id           0.000000e+00
         failed_flag            0.000000e+00
```

```
phi                      3.057608e+06
meas_phi                 3.057608e+06
regul_phi                0.000000e+00
calflux                  3.057602e+06
flx_in-out               0.000000e+00
flx_storage              0.000000e+00
flout                    0.000000e+00
vol_recharg              0.000000e+00
vol_storage              0.000000e+00
flaqx                    0.000000e+00
vol_constan              0.000000e+00
vol_percent              0.000000e+00
vol_drains               0.000000e+00
flx_total                0.000000e+00
hds                      0.000000e+00
obgnme                   0.000000e+00
vol_stream_              0.000000e+00
flx_drains               0.000000e+00
calhead                  5.723258e+00
flx_wells                0.000000e+00
flx_constan              0.000000e+00
flx_stream_              0.000000e+00
vol_wells                0.000000e+00
vol_in-out               0.000000e+00
flx_recharg              0.000000e+00
flx_percent              0.000000e+00
vol_total                0.000000e+00
fa_0_19791230           -8.396900e+01
                             ...
hds_02_039_010_000       3.255568e+01
hds_02_039_010_001       3.252518e+01
hds_02_039_011_000       3.253960e+01
hds_02_039_011_001       3.251804e+01
hds_02_039_012_000       3.253343e+01
hds_02_039_012_001       3.251686e+01
hds_02_039_013_000       3.253542e+01
hds_02_039_013_001       3.252187e+01
hds_02_039_014_000       3.254475e+01
hds_02_039_014_001       3.253363e+01
vol_constan_19791230     0.000000e+00
vol_constan_19801229     0.000000e+00
vol_drains_19791230     -2.527600e+06
vol_drains_19801229     -2.644955e+06
vol_in-out_19791230     -2.431320e+05
vol_in-out_19801229     -2.427030e+05
vol_percent_19791230    -1.930000e+00
vol_percent_19801229    -1.810000e+00
vol_recharg_19791230     1.208895e+07
```

```
vol_recharg_19801229    1.235586e+07
vol_storage_19791230    2.415926e+05
vol_storage_19801229    6.609141e+05
vol_stream__19791230   -5.542931e+06
vol_stream__19801229   -5.544508e+06
vol_total_19791230     -2.431320e+05
vol_total_19801229     -2.427030e+05
vol_wells_19791230     -4.503142e+06
vol_wells_19801229     -5.070016e+06
part_status             2.000000e+00
part_time               2.025333e+03
Name: 0, Length: 4465, dtype: float64
```

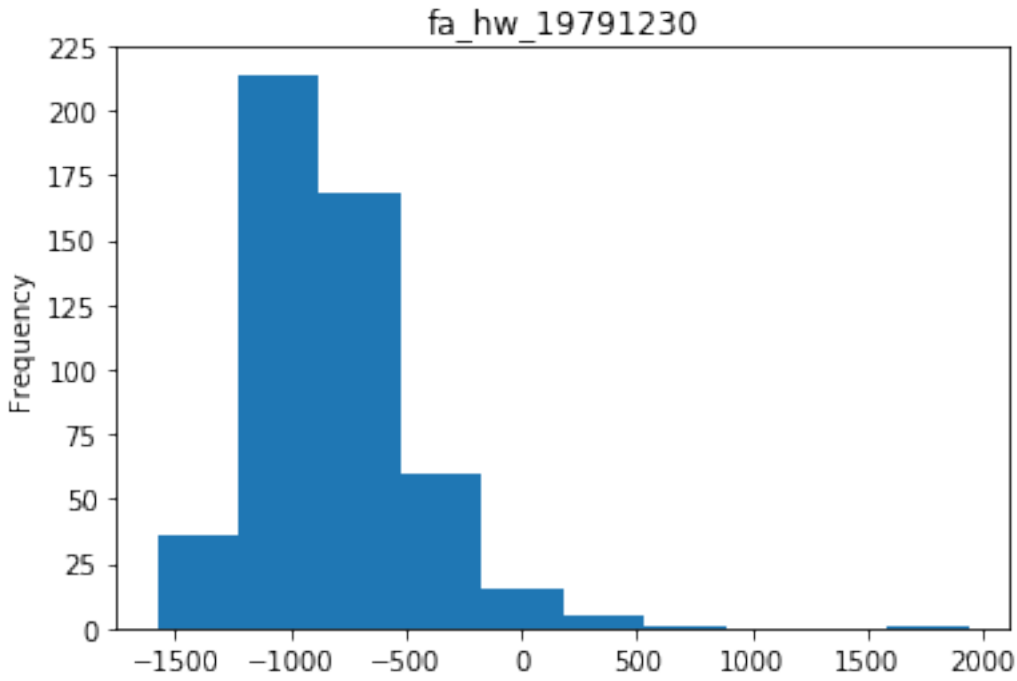
1.0.5 confirm which quantities were identified as forecasts

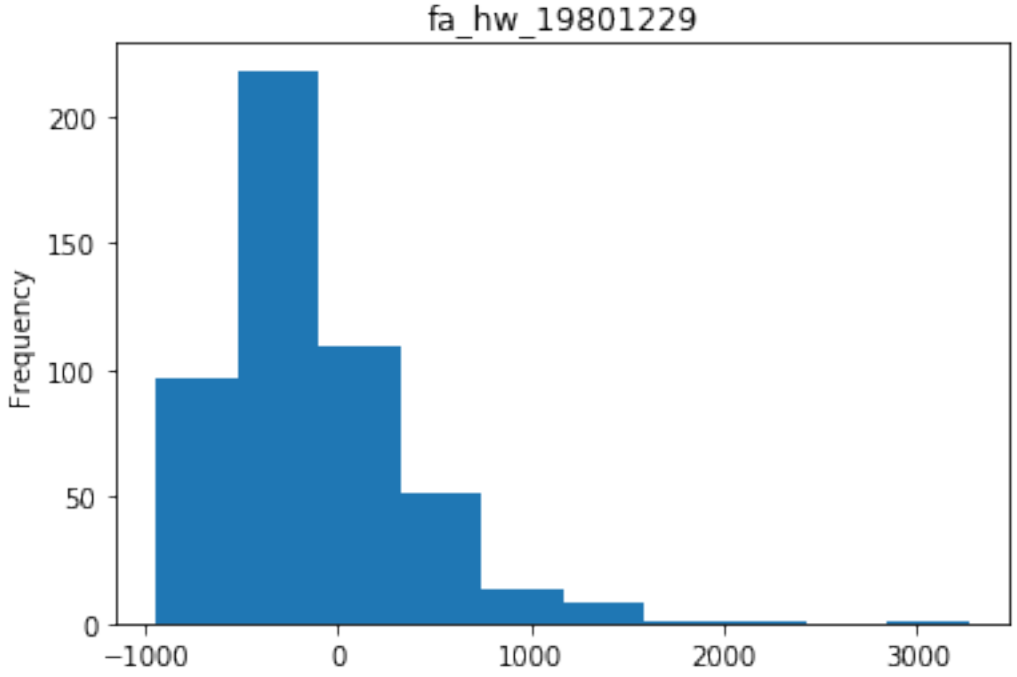
```
In [11]: fnames = pst.pestpp_options["forecasts"].split(',')
         fnames
```

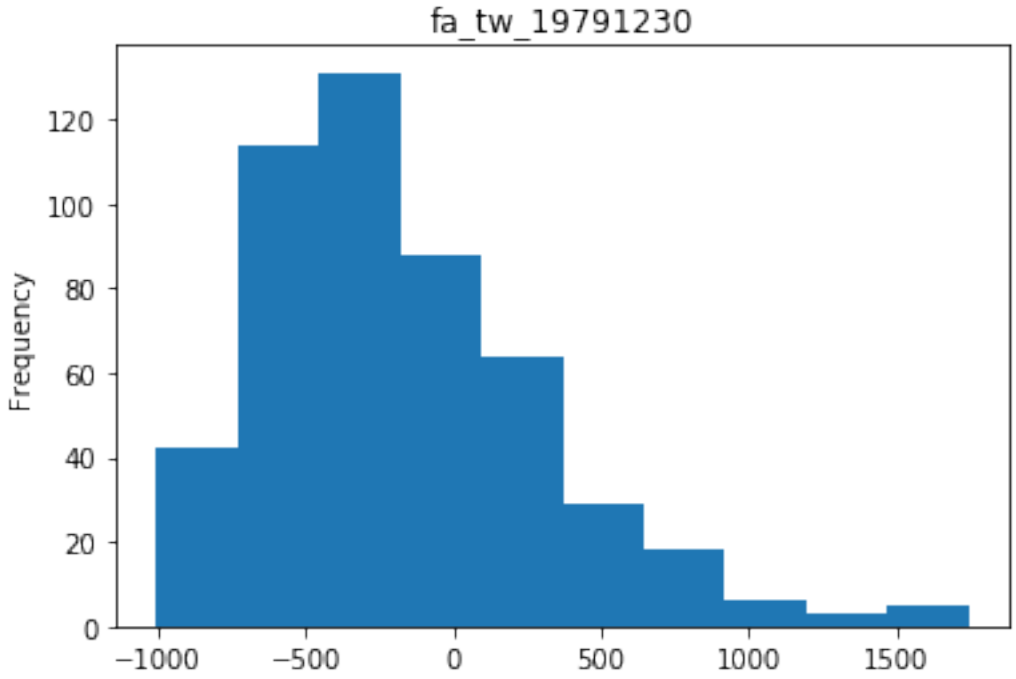
```
Out[11]: ['fa_hw_19791230',
          'fa_hw_19801229',
          'fa_tw_19791230',
          'fa_tw_19801229',
          'hds_00_013_002_000',
          'hds_00_013_002_001',
          'part_time',
          'part_status']
```

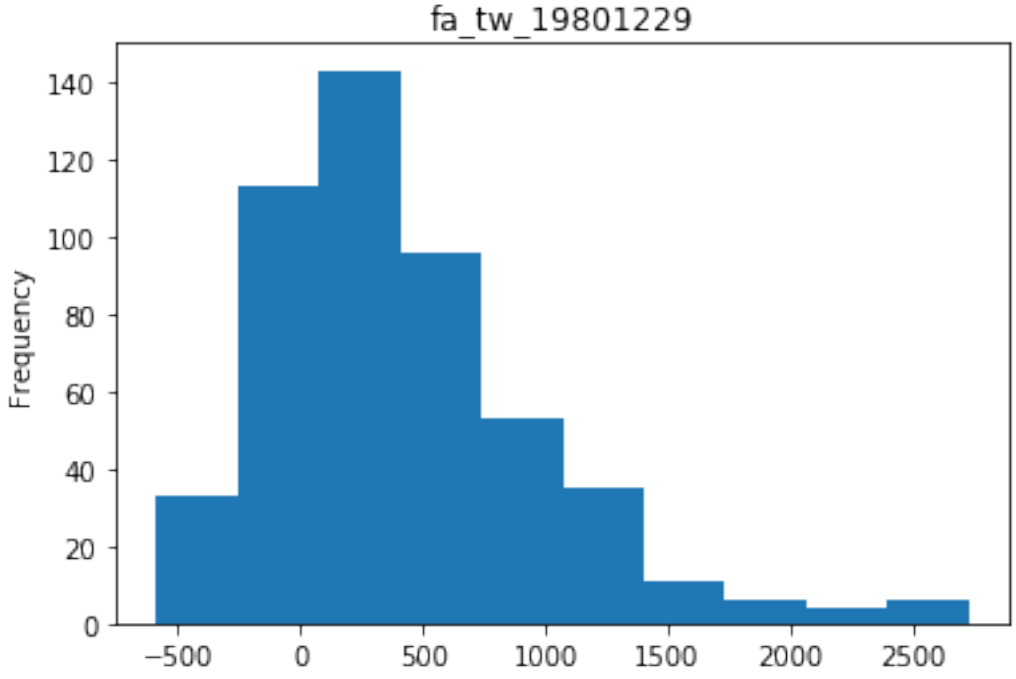
1.0.6 now we can plot the distributions of each forecast

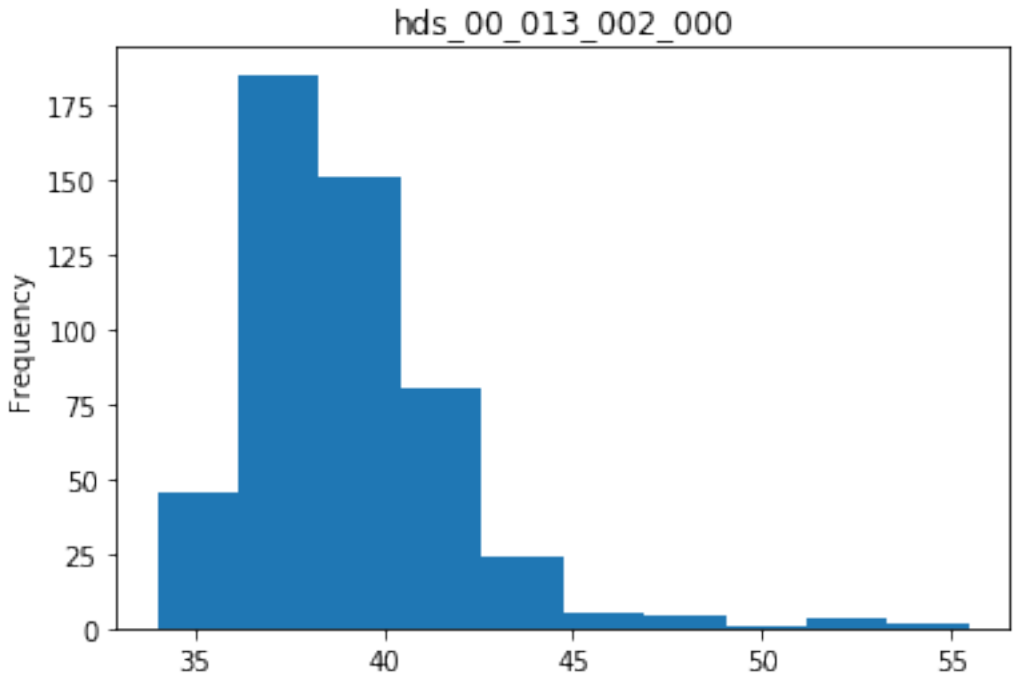
```
In [12]: for forecast in fnames:
             plt.figure()
             ax = obs_df.loc[:,forecast].plot(kind="hist")
             ax.set_title(forecast)
         plt.show()
```

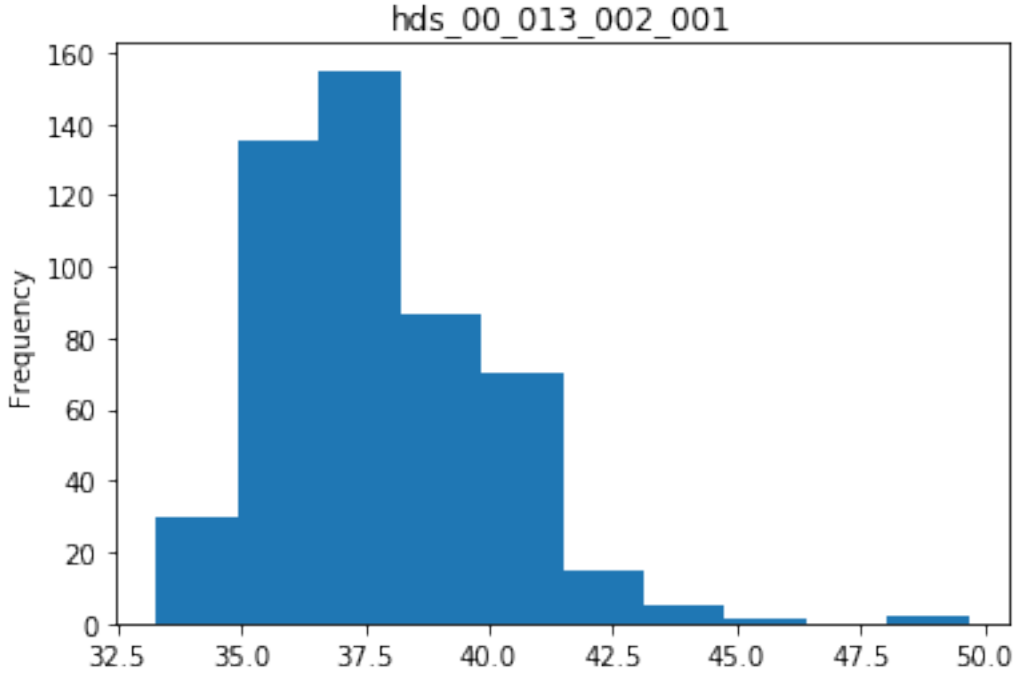

fa_hw_19791230
Frequency
225
200
175
150
125
100
75
50
25
0
−1500
−1000
−500
0
500
1000
1500
2000

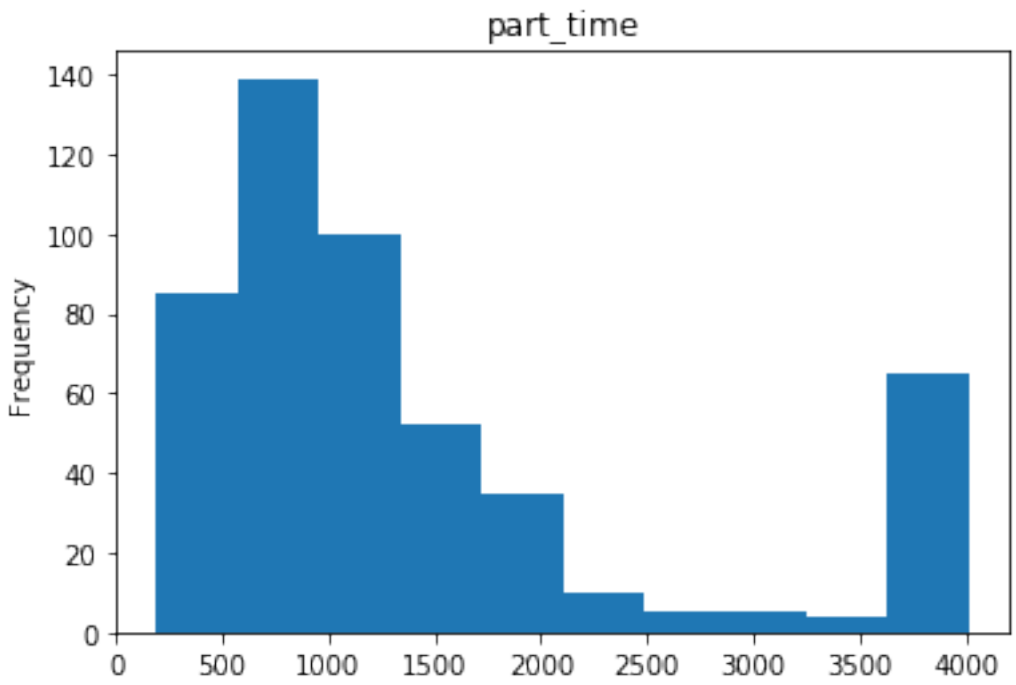

fa_hw_19801229
Frequency
200
150
100
50
0
−1000
0
1000
2000
3000

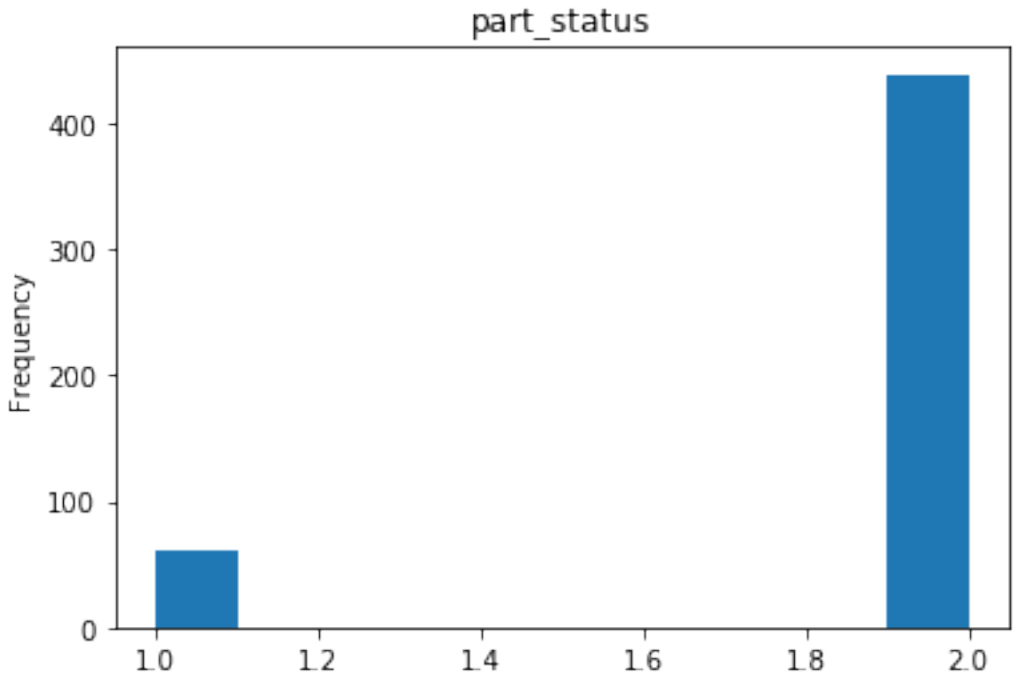

fa_tw_19791230
Frequency
120
100
80
60
40
20
0
−1000
−500
0
500
1000
1500


fa_tw_19801229
Frequency
140
120
100
80
60
40
20
0
−500
0
500
1000
1500
2000
2500


hds_00_013_002_000
Frequency
175
150
125
100
75
50
25
0
35
40
45
50
55

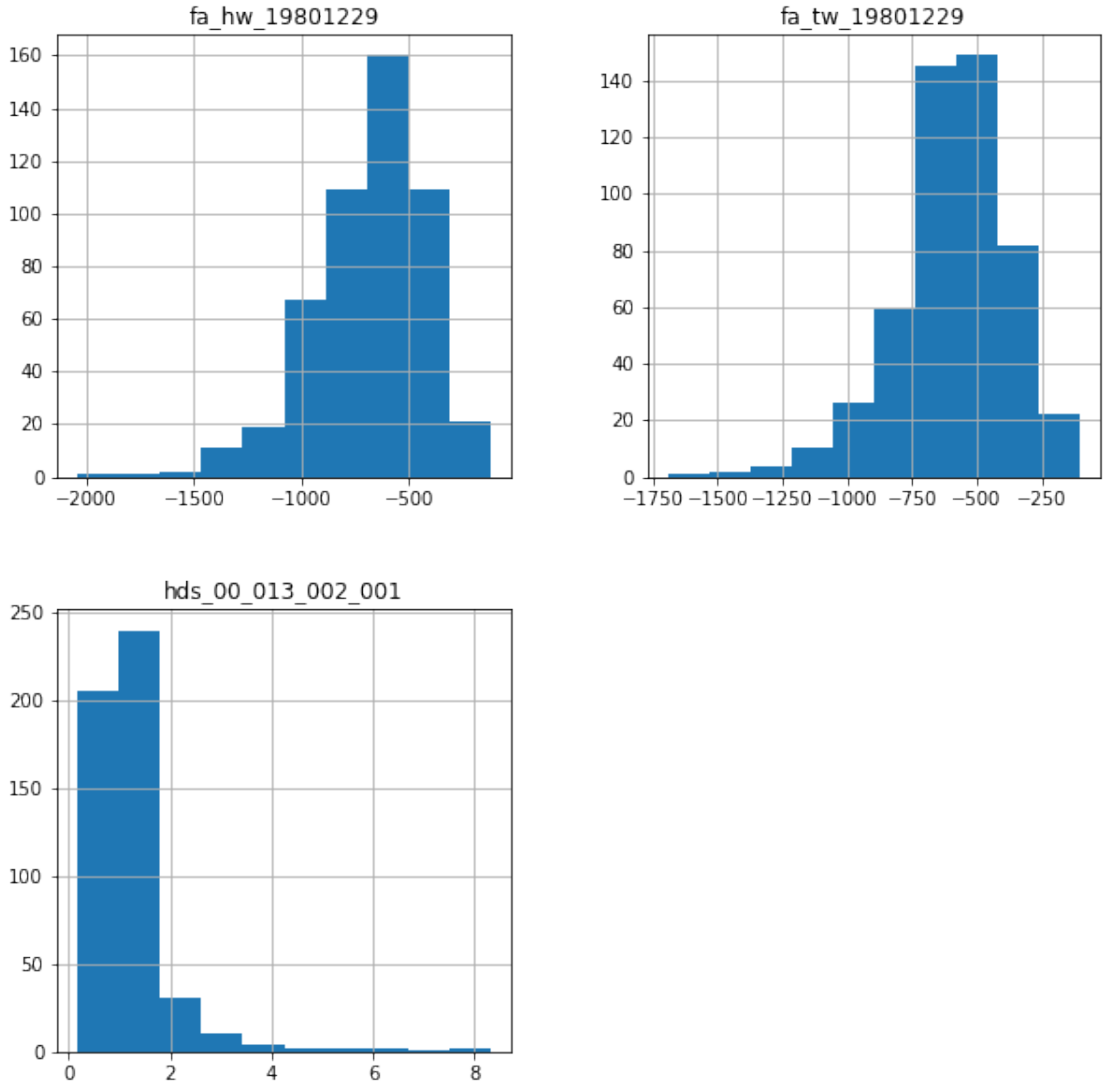

hds_00_013_002_001
Frequency
160
140
120
100
80
60
40
20
0
32.5
35.0
37.5
40.0
42.5
45.0
47.5
50.0





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 [13]: 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 [14]: # choose the realization with a low historic gw to sw headwater flux
         #hist_swgw = obs_df.loc[:,"fa_hw_19791230"].sort_values()
         hist_swgw = obs_df.loc[:,"part_time"].sort_values()
         idx = hist_swgw.index[20]
         idx
         hist_swgw
```

```
Out[14]: run_id
         342     192.0600
         262     211.8256
         97      216.6293
         233     226.0260
         4       251.3049
         91      263.8806
         17      273.0211
         397     290.3943
         94      294.2234
         77      301.3943
         473     303.4344
         197     326.5027
         57      336.2830
         69      338.4649
         361     349.9784
         477     351.6884
         239     362.2070
         466     362.7992
         222     362.9305
         343     376.1986
         307     377.0269
         25      383.8937
         136     388.1931
         92      390.5105
         451     392.2232
         83      392.7031
         378     401.5639
         144     403.3770
         155     407.3019
         329     407.3465
                  ...
         181    4015.0000
         177    4015.0000
         412    4015.0000
         411    4015.0000
         410    4015.0000
         176    4015.0000
```

```
393    4015.0000
183    4015.0000
132    4015.0000
129    4015.0000
128    4015.0000
75     4015.0000
246    4015.0000
79     4015.0000
85     4015.0000
240    4015.0000
90     4015.0000
355    4015.0000
356    4015.0000
220    4015.0000
106    4015.0000
107    4015.0000
110    4015.0000
365    4015.0000
113    4015.0000
214    4015.0000
368    4015.0000
430    4015.0000
65     4015.0000
302    4015.0000
Name: part_time, Length: 500, dtype: float64
```

```
In [15]: obs_df.loc[idx,pst.nnz_obs_names]
```

```
Out[15]: fo_39_19791230        10874.000000
         hds_00_002_009_000       36.574207
         hds_00_002_015_000       35.057098
         hds_00_003_008_000       36.682930
         hds_00_009_001_000       40.124496
         hds_00_013_010_000       35.340054
         hds_00_015_016_000       34.673721
         hds_00_021_010_000       36.465126
         hds_00_022_015_000       34.842373
         hds_00_024_004_000       39.561016
         hds_00_026_006_000       38.190109
         hds_00_029_015_000       34.329807
         hds_00_033_007_000       35.755814
         hds_00_034_010_000       34.562611
         Name: 307, dtype: float64
```

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

```
In [16]: obs_df.loc[idx,fnames]
```

```
Out[16]: fa_hw_19791230       -1165.197110
         fa_hw_19801229        -154.891060
```

```
fa_tw_19791230        -505.492900
fa_tw_19801229          38.589600
hds_00_013_002_000      41.254593
hds_00_013_002_001      40.049156
part_time              377.026900
part_status              2.000000
Name: 307, 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 [17]: 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"] = 1.0
```

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 [18]: 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[18]: obsnme
         fo_39_19791230        1.764052
         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

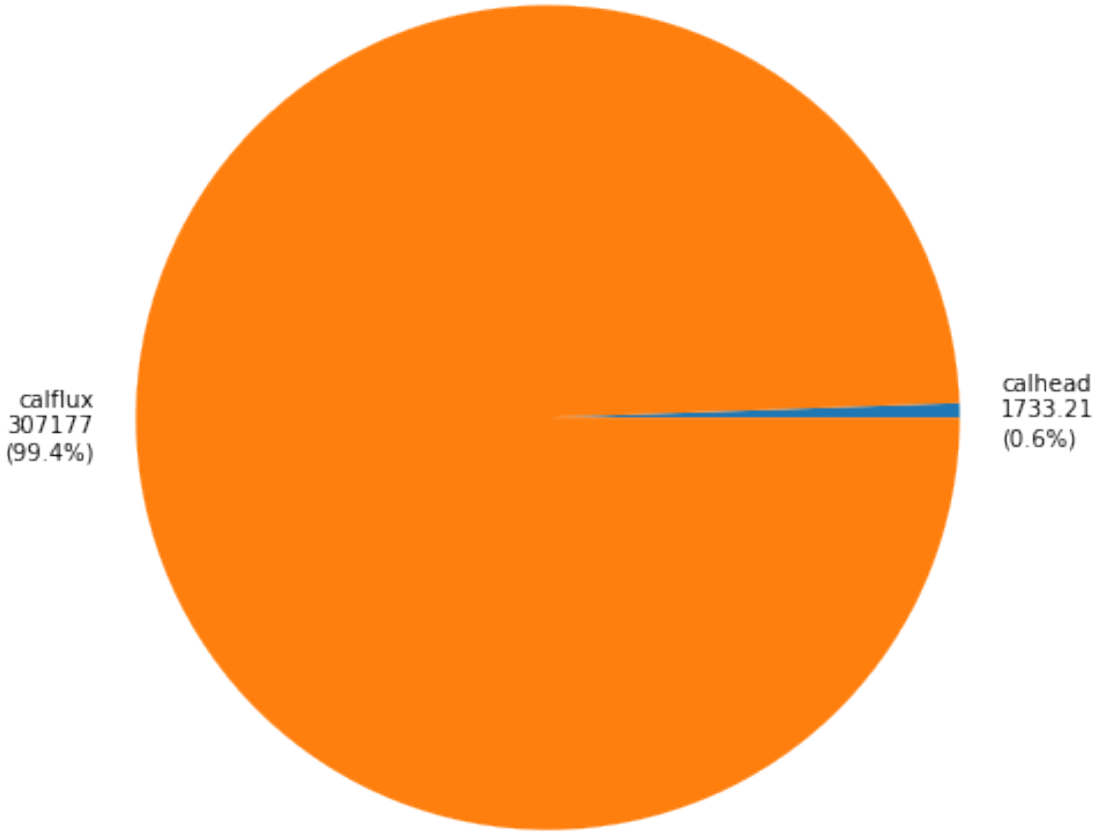
```
In [19]: pst.write(os.path.join(t_d,"freyberg.pst"))
         pyemu.os_utils.run("pestpp-ies freyberg.pst",cwd=t_d)
```

```
noptmax:0, npar_adj:14819, nnz_obs:14
```

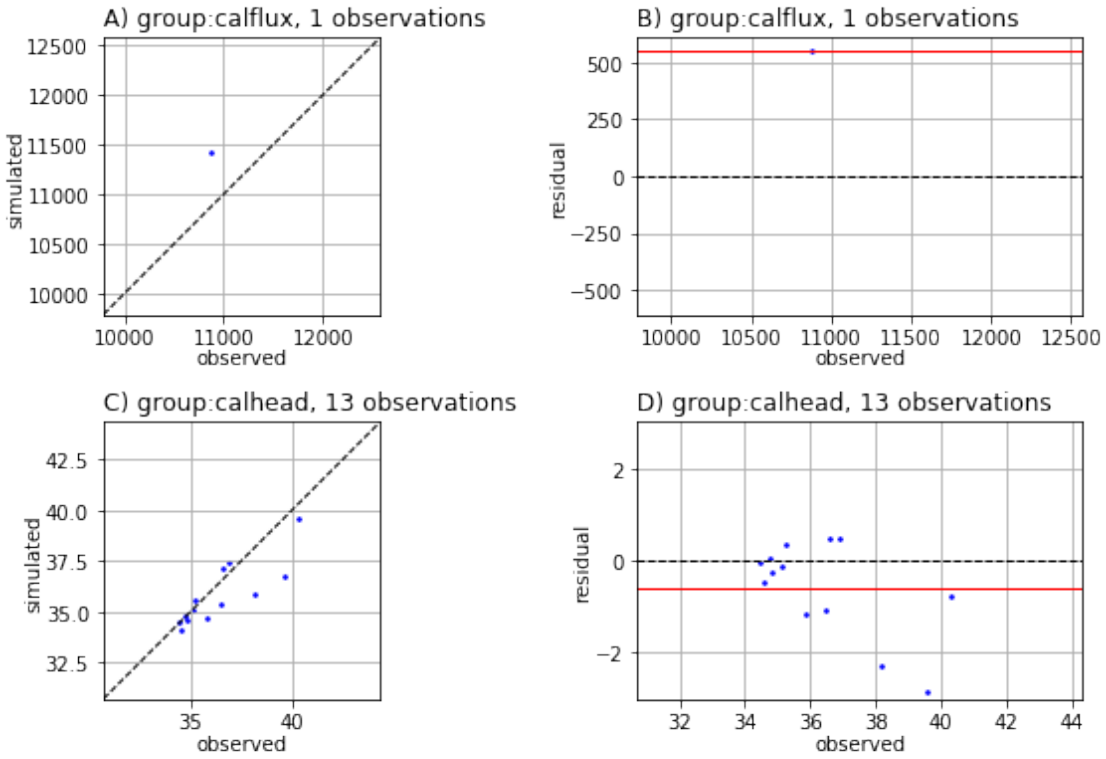
Now we can read in the results and make some figures showing residuals and the balance of the objective function

```
In [20]: 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,:]
```

```
308910.69418201654
Here are the non-zero weighted observation names


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



<Figure size 576x756 with 0 Axes>



Out[20]:

name	group	measured	modelled	\
name				
fo_39_19791230	fo_39_19791230	calflux	10875.764052	11430.000000

```
hds_00_002_009_000  hds_00_002_009_000  calhead     36.614223     37.107498
hds_00_002_015_000  hds_00_002_015_000  calhead     35.154972     35.045185
hds_00_003_008_000  hds_00_003_008_000  calhead     36.907019     37.397289
hds_00_009_001_000  hds_00_009_001_000  calhead     40.311252     39.546417
hds_00_013_010_000  hds_00_013_010_000  calhead     35.242326     35.571774
hds_00_015_016_000  hds_00_015_016_000  calhead     34.768730     34.835716
hds_00_021_010_000  hds_00_021_010_000  calhead     36.449990     35.386250
hds_00_022_015_000  hds_00_022_015_000  calhead     34.832051     34.577492
hds_00_024_004_000  hds_00_024_004_000  calhead     39.602076     36.760464
hds_00_026_006_000  hds_00_026_006_000  calhead     38.204514     35.896149
hds_00_029_015_000  hds_00_029_015_000  calhead     34.475235     34.453842
hds_00_033_007_000  hds_00_033_007_000  calhead     35.831917     34.678810
hds_00_034_010_000  hds_00_034_010_000  calhead     34.574778     34.118073
```

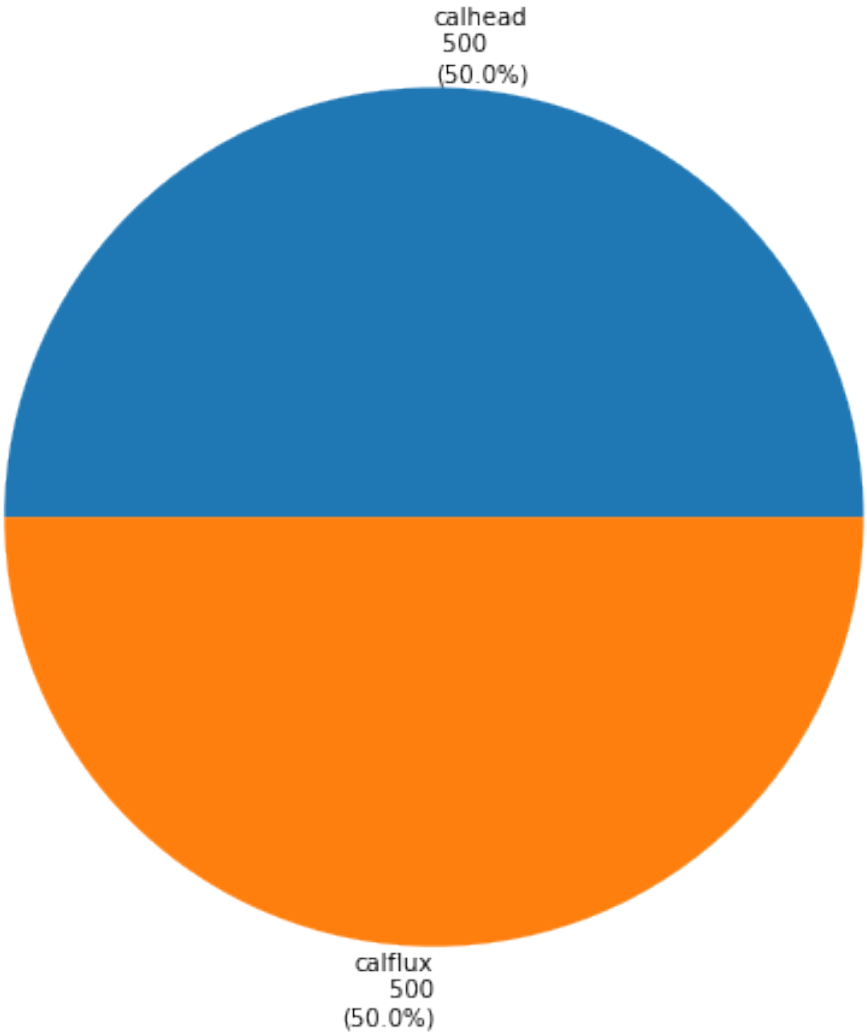
name	residual	weight
fo_39_19791230	-554.235948	1.0
hds_00_002_009_000	-0.493275	10.0
hds_00_002_015_000	0.109787	10.0
hds_00_003_008_000	-0.490270	10.0
hds_00_009_001_000	0.764835	10.0
hds_00_013_010_000	-0.329448	10.0
hds_00_015_016_000	-0.066986	10.0
hds_00_021_010_000	1.063741	10.0
hds_00_022_015_000	0.254559	10.0
hds_00_024_004_000	2.841612	10.0
hds_00_026_006_000	2.308365	10.0
hds_00_029_015_000	0.021392	10.0
hds_00_033_007_000	1.153107	10.0
hds_00_034_010_000	0.456706	10.0

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 [21]:
```
pc = pst.phi_components
#target = {"calflux":0.3 * pc["calhead"]}
target = {"calhead":500,"calflux":500}
pst.adjust_weights(obsgrp_dict=target)
pst.plot(kind='phi_pie')
```

Out[21]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x11f03b6d8>
```



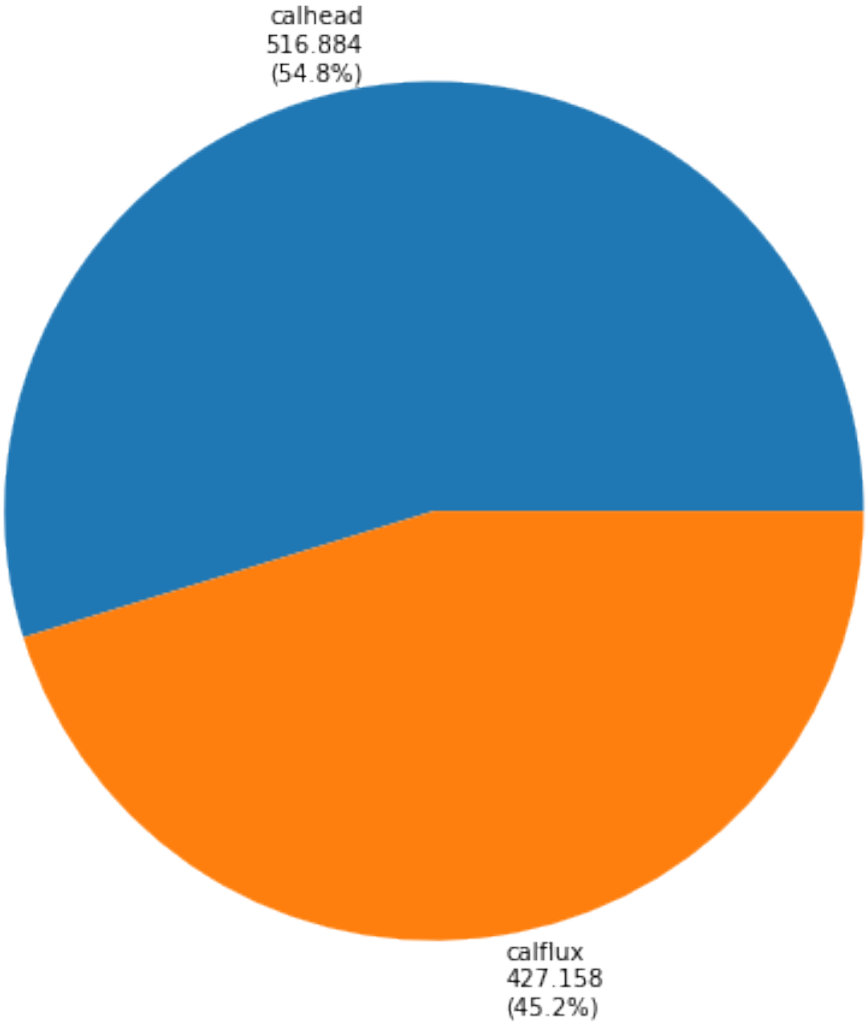
Lets see what the new flux observation weight is:

```
In [22]: pst.observation_data.loc[pst.nnz_obs_names,"weight"]

Out[22]: obsnme
         fo_39_19791230        0.040345
         hds_00_002_009_000    5.371056
         hds_00_002_015_000    5.371056
         hds_00_003_008_000    5.371056
         hds_00_009_001_000    5.371056
         hds_00_013_010_000    5.371056
         hds_00_015_016_000    5.371056
         hds_00_021_010_000    5.371056
         hds_00_022_015_000    5.371056
```

```
hds_00_024_004_000    5.371056
hds_00_026_006_000    5.371056
hds_00_029_015_000    5.371056
hds_00_033_007_000    5.371056
hds_00_034_010_000    5.371056
Name: weight, dtype: float64
```

Now, for some super trickery: since we changed the weight, we need to generate the observation noise using these new weights for the error model (so meta!)

```
In [23]: obs = pst.observation_data
         np.random.seed(seed=0)
         snd = np.random.randn(pst.nnz_obs)
         noise = snd * 1./obs.loc[pst.nnz_obs_names,"weight"]
         obs.loc[:,"obsval"] = obs_df.loc[idx,pst.obs_names]
         pst.observation_data.loc[noise.index,"obsval"] += noise
         noise

Out[23]: obsnme
         fo_39_19791230        43.724128
         hds_00_002_009_000     0.074503
         hds_00_002_015_000     0.182225
         hds_00_003_008_000     0.417217
         hds_00_009_001_000     0.347708
         hds_00_013_010_000    -0.181953
         hds_00_015_016_000     0.176890
         hds_00_021_010_000    -0.028180
         hds_00_022_015_000    -0.019218
         hds_00_024_004_000     0.076447
         hds_00_026_006_000     0.026818
         hds_00_029_015_000     0.270761
         hds_00_033_007_000     0.141692
         hds_00_034_010_000     0.022654
         Name: weight, dtype: float64

In [24]: pst.write(os.path.join(t_d,"freyberg.pst"))
         pyemu.os_utils.run("pestpp-ies freyberg.pst",cwd=t_d)
         pst = pyemu.Pst(os.path.join(t_d,"freyberg.pst"))
         print(pst.phi)
         pst.plot(kind='phi_pie')
         plt.show()

noptmax:0, npar_adj:14819, nnz_obs:14
944.0423318749266
```



Whew! confused yet? Ok, let's leave all this confusion behind...its mostly academic, just to make sure we are using weights that are in harmony with the noise we added to the truth...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 [25]: 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"))

noptmax:0, npar_adj:14819, nnz_obs:14
```

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 [26]: 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,:]
```

```
17.528847206992
```

Out[26]:

	name	group	measured	modelled	residual	weight
name						
fo_39_19791230	fo_39_19791230	calflux	10917.724128	10874.000000	43.724128	0.040345
hds_00_002_009_000	hds_00_002_009_000	calhead	36.648710	36.574207	0.074503	5.371056
hds_00_002_015_000	hds_00_002_015_000	calhead	35.239323	35.057098	0.182225	5.371056
hds_00_003_008_000	hds_00_003_008_000	calhead	37.100147	36.682930	0.417217	5.371056
hds_00_009_001_000	hds_00_009_001_000	calhead	40.472204	40.124496	0.347708	5.371056
hds_00_013_010_000	hds_00_013_010_000	calhead	35.158101	35.340054	-0.181953	5.371056
hds_00_015_016_000	hds_00_015_016_000	calhead	34.850612	34.673721	0.176890	5.371056
hds_00_021_010_000	hds_00_021_010_000	calhead	36.436946	36.465126	-0.028180	5.371056
hds_00_022_015_000	hds_00_022_015_000	calhead	34.823155	34.842373	-0.019218	5.371056
hds_00_024_004_000	hds_00_024_004_000	calhead	39.637463	39.561016	0.076447	5.371056
hds_00_026_006_000	hds_00_026_006_000	calhead	38.216928	38.190109	0.026818	5.371056
hds_00_029_015_000	hds_00_029_015_000	calhead	34.600568	34.329807	0.270761	5.371056
hds_00_033_007_000	hds_00_033_007_000	calhead	35.897506	35.755814	0.141692	5.371056
hds_00_034_010_000	hds_00_034_010_000	calhead	34.585264	34.562611	0.022654	5.371056

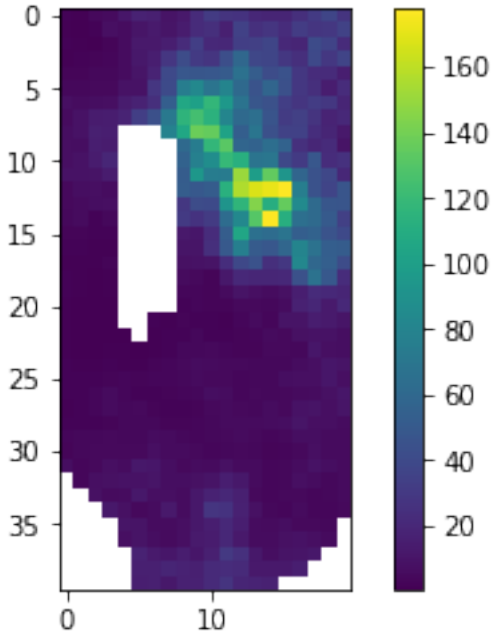
The residual should be exactly the noise values from above.Lets load the model (that was just run using the true pars) and check some things

```
In [27]: m = flopy.modflow.Modflow.load("freyberg.nam",model_ws=m_d)
```

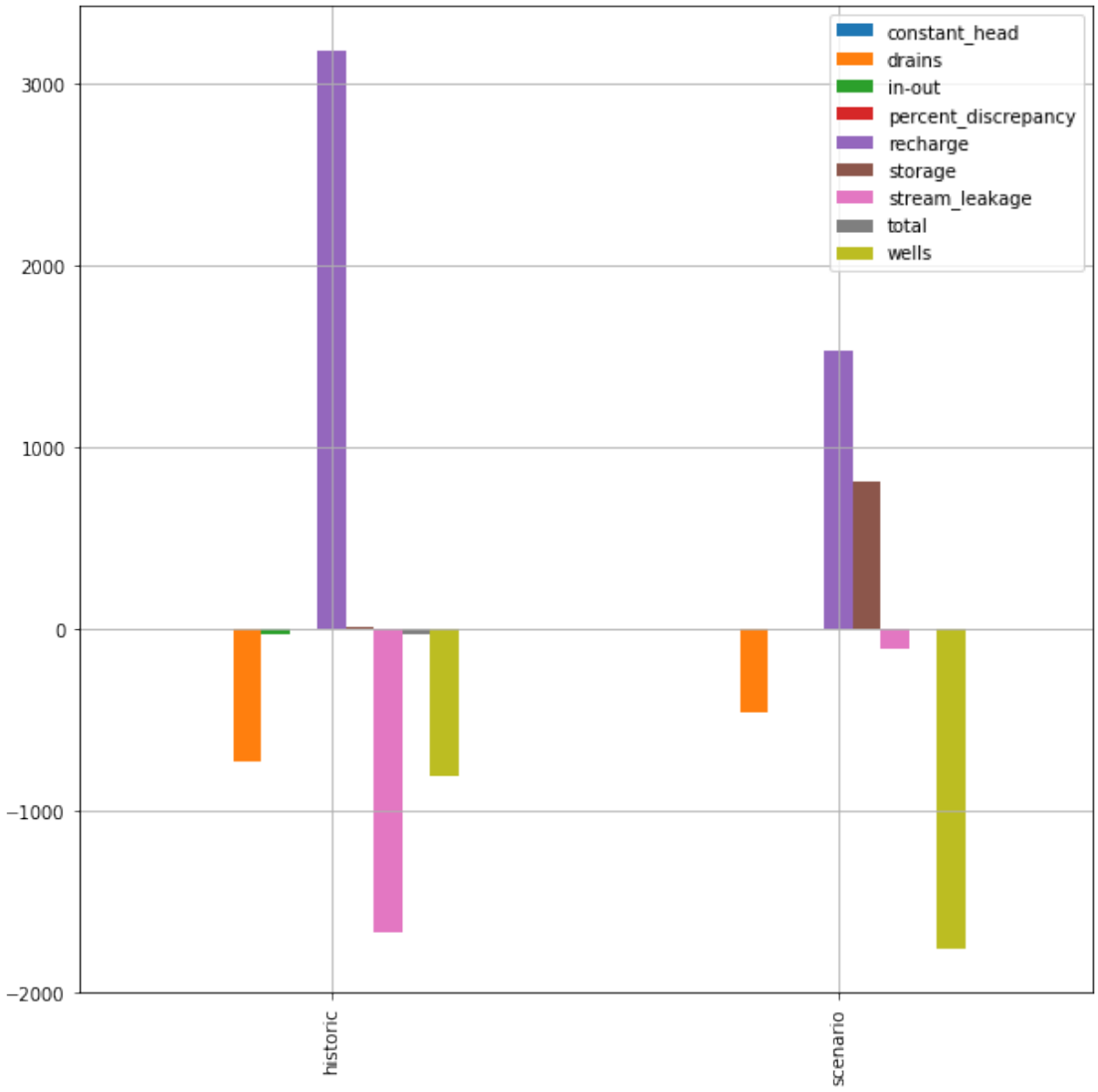
```
In [28]: a = m.upw.hk[0].array
         #a = m.rch.rech[0].array
```

```
        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()
```

```
0.6367541 177.6865
```



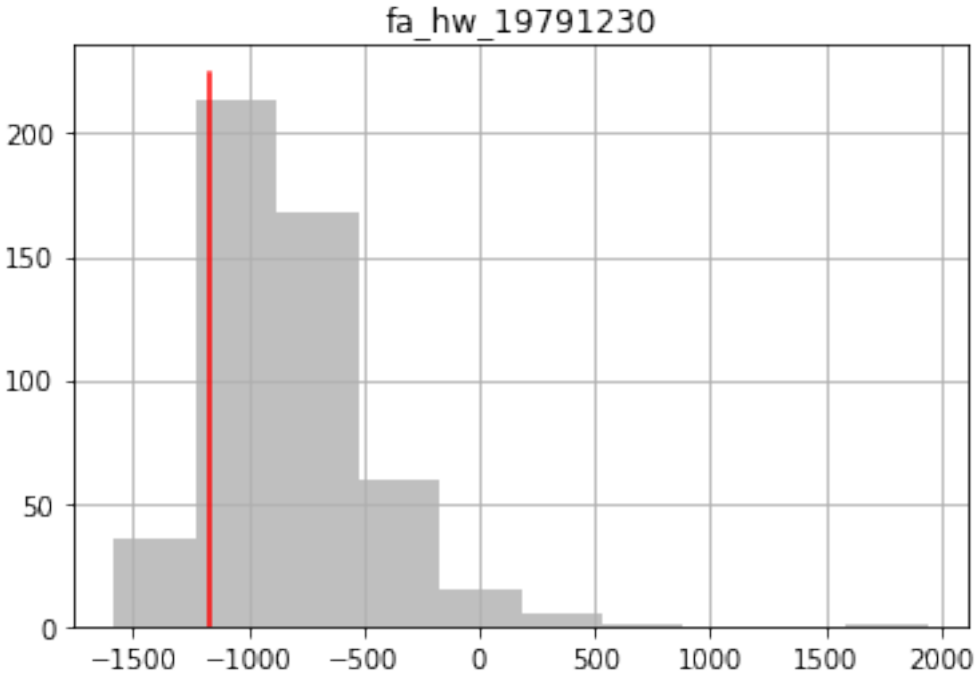

```
In [29]: lst = flopy.utils.MfListBudget(os.path.join(m_d,"freyberg.list"))
         df = lst.get_dataframes(diff=True)[0]
         ax = df.plot(kind="bar",figsize=(10,10), grid=True)
         a = ax.set_xticklabels(["historic","scenario"],rotation=90)
         plt.show(ax)
```

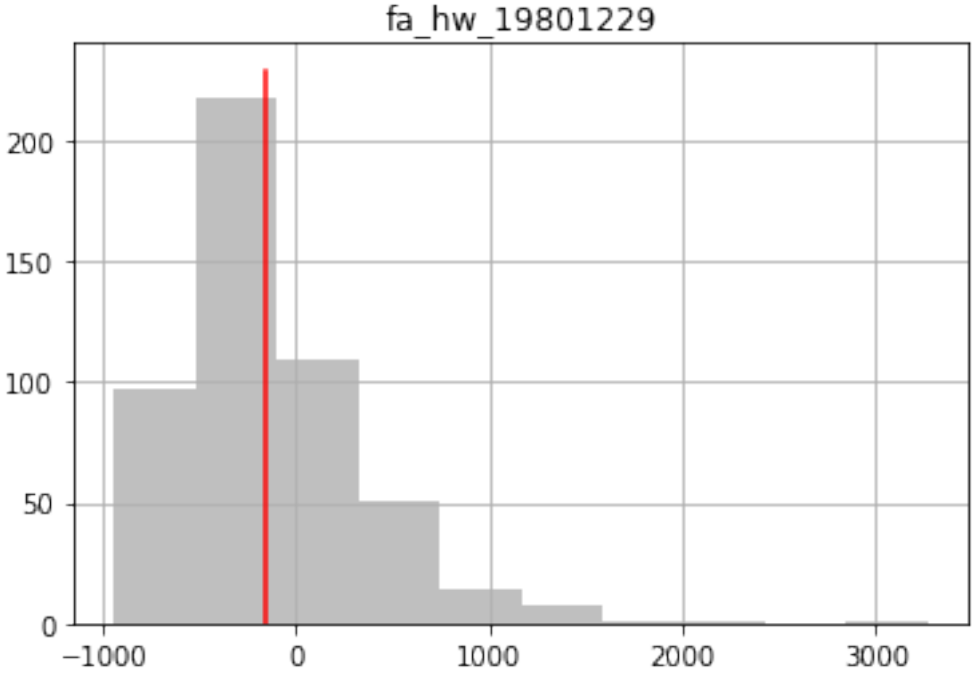


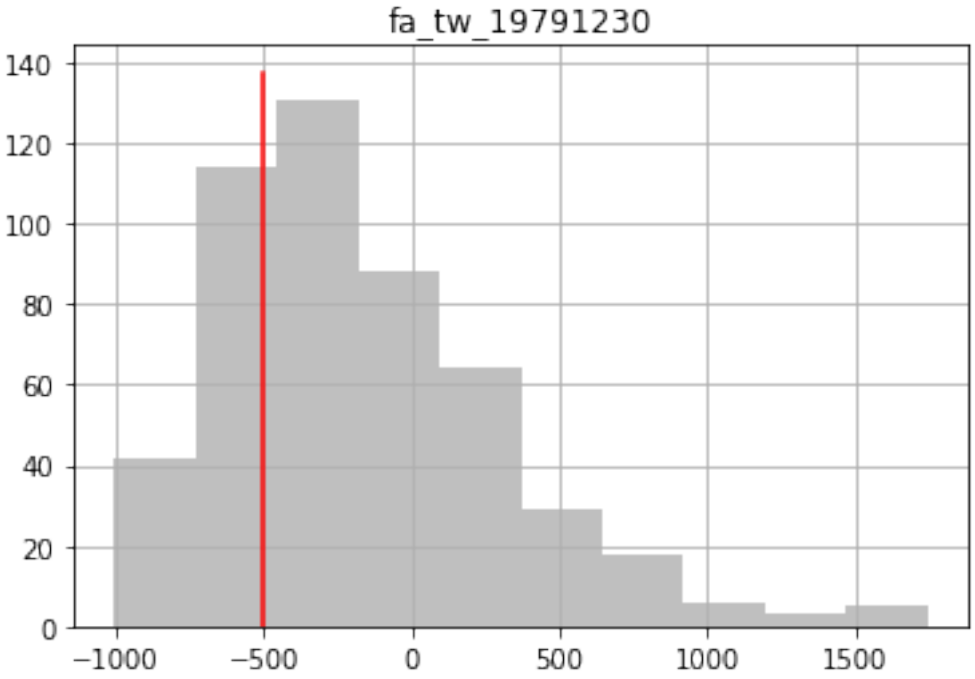
1.0.8 see how our existing observation ensemble compares to the truth

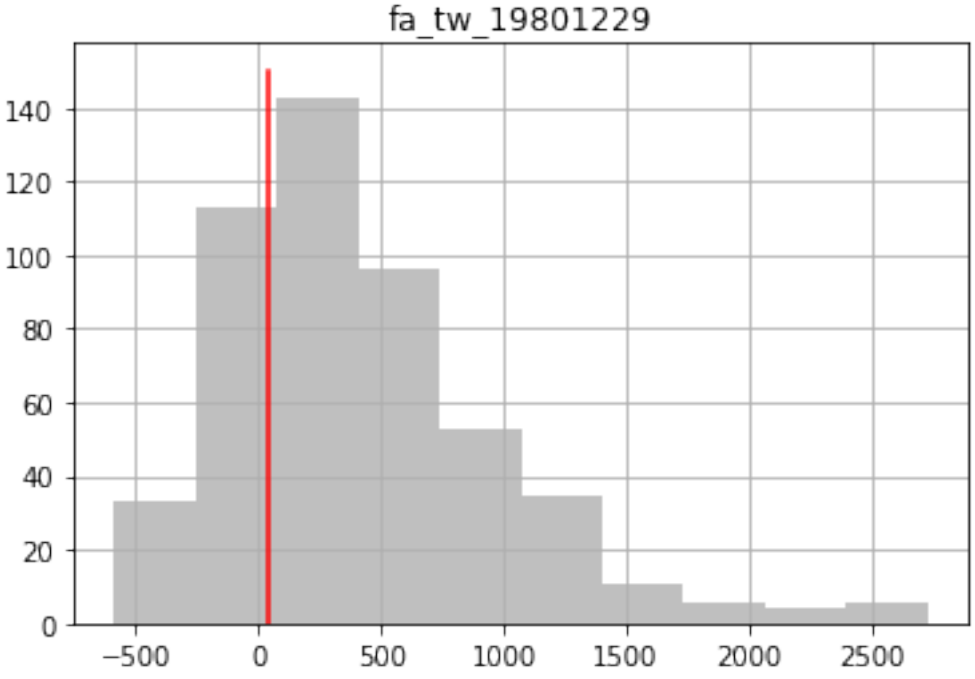
forecasts:

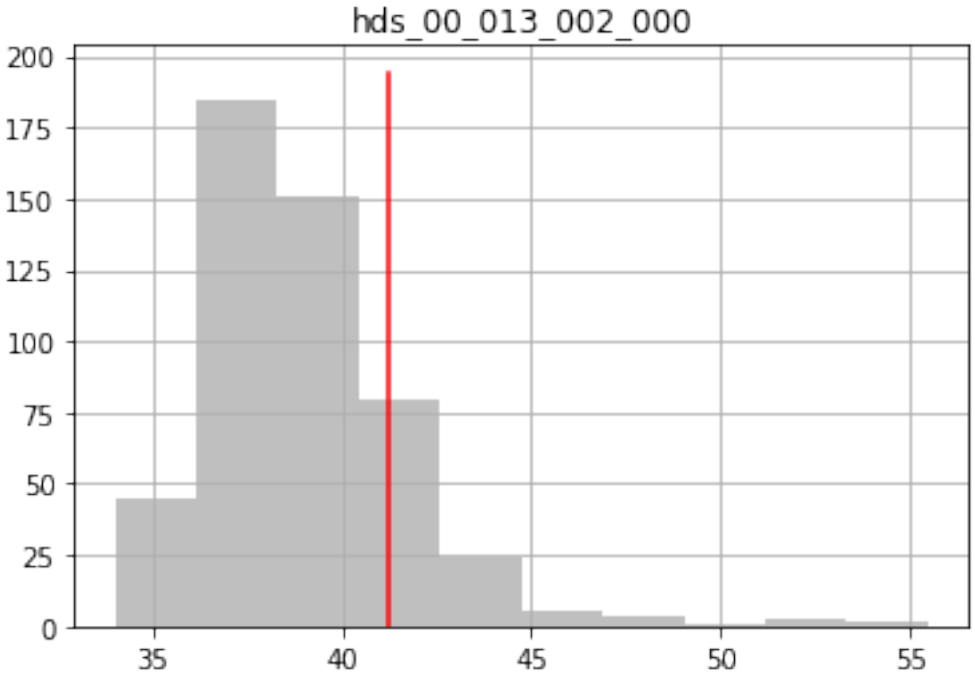
```
In [30]: obs = pst.observation_data
         plt.figure()
         for forecast in fnames:
             ax = plt.subplot(111)
             obs_df.loc[:,forecast].hist(ax=ax,color="0.5",alpha=0.5)
             ax.plot([obs.loc[forecast,"obsval"],obs.loc[forecast,"obsval"]],ax.get_ylim(),"r"
             ax.set_title(forecast)
             plt.show()
```

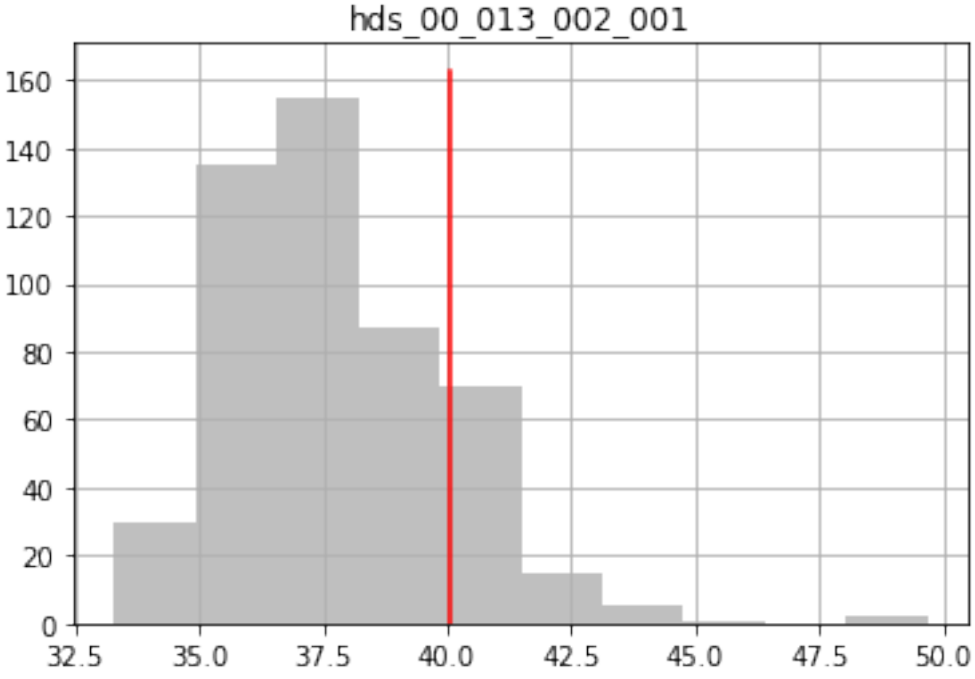

fa_hw_19791230
200
150
100
50
0
−1500
−1000
−500
0
500
1000
1500
2000

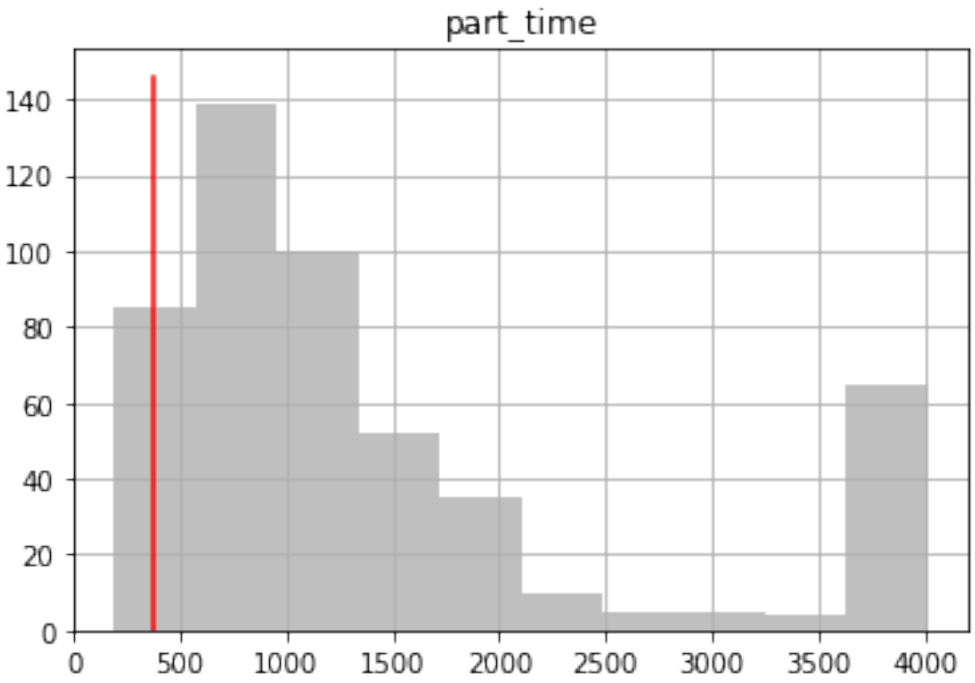

fa_hw_19801229
200
150
100
50
0
−1000
0
1000
2000
3000

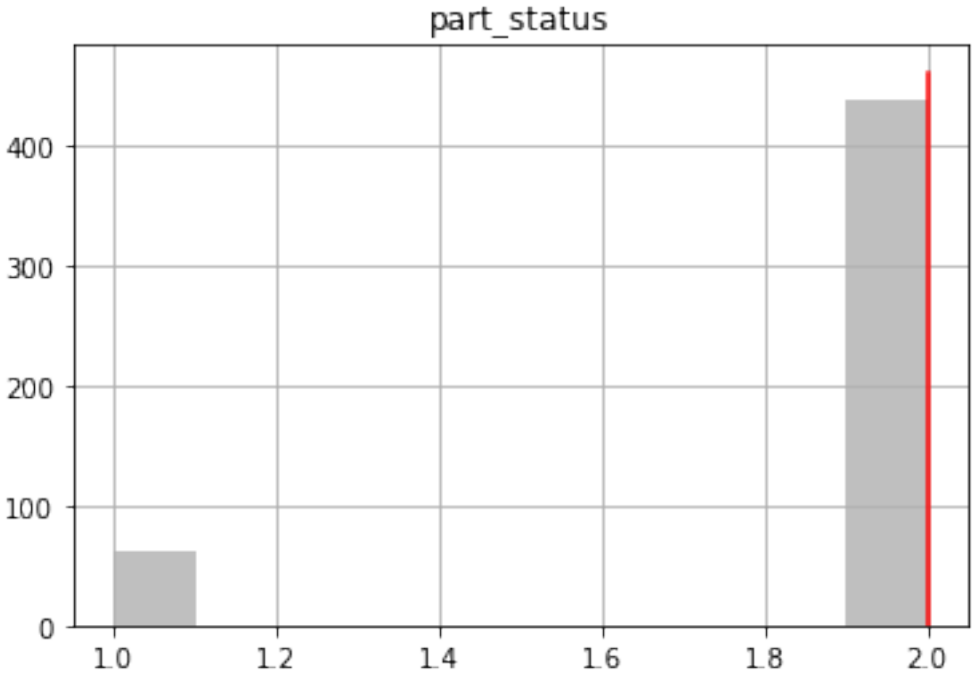

fa_tw_19791230
140
120
100
80
60
40
20
0
−1000
−500
0
500
1000
1500


fa_tw_19801229
140
120
100
80
60
40
20
0
−500
0
500
1000
1500
2000
2500


hds_00_013_002_000
200
175
150
125
100
75
50
25
0
35
40
45
50
55

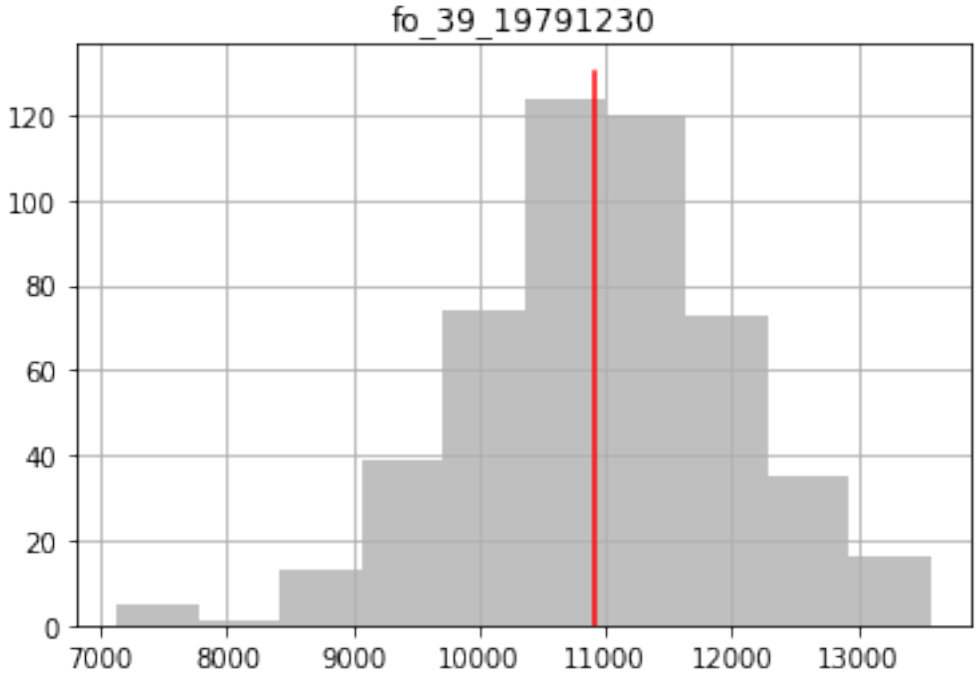

hds_00_013_002_001
160
140
120
100
80
60
40
20
0
32.5
35.0
37.5
40.0
42.5
45.0
47.5
50.0

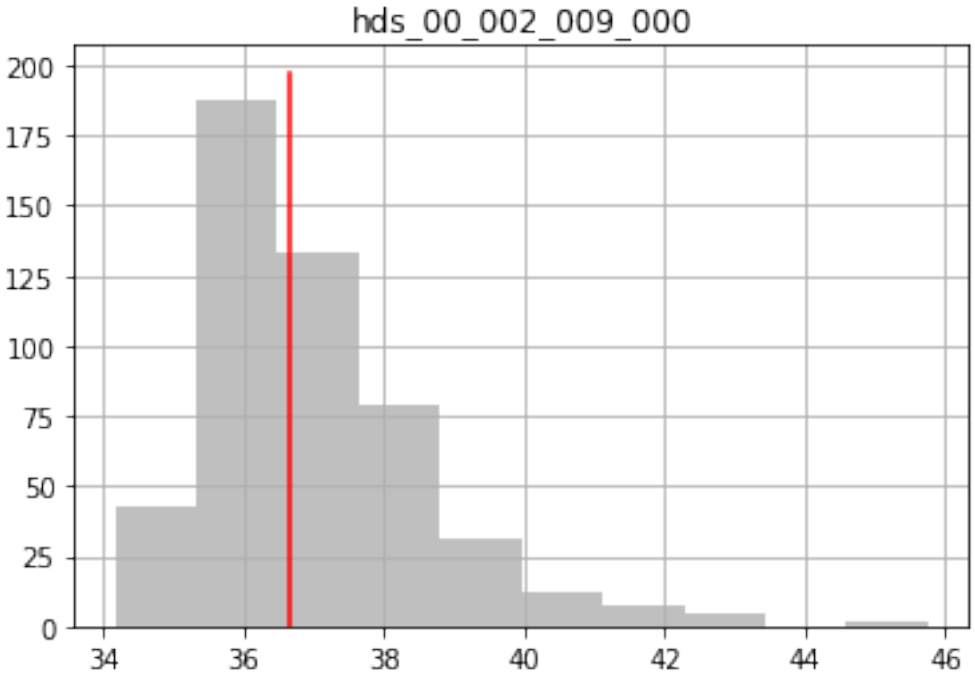


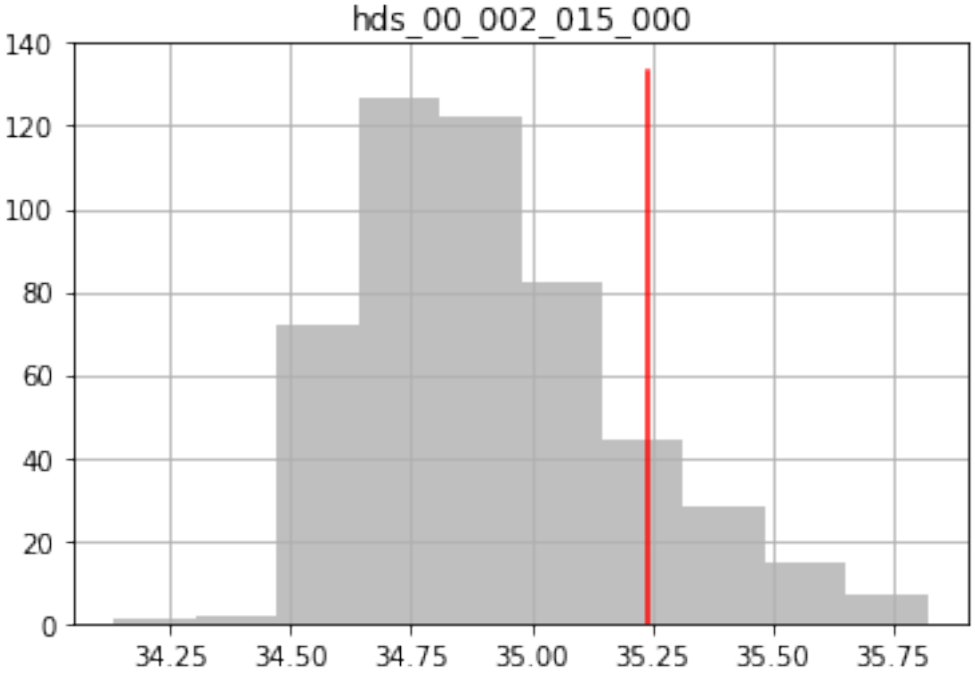


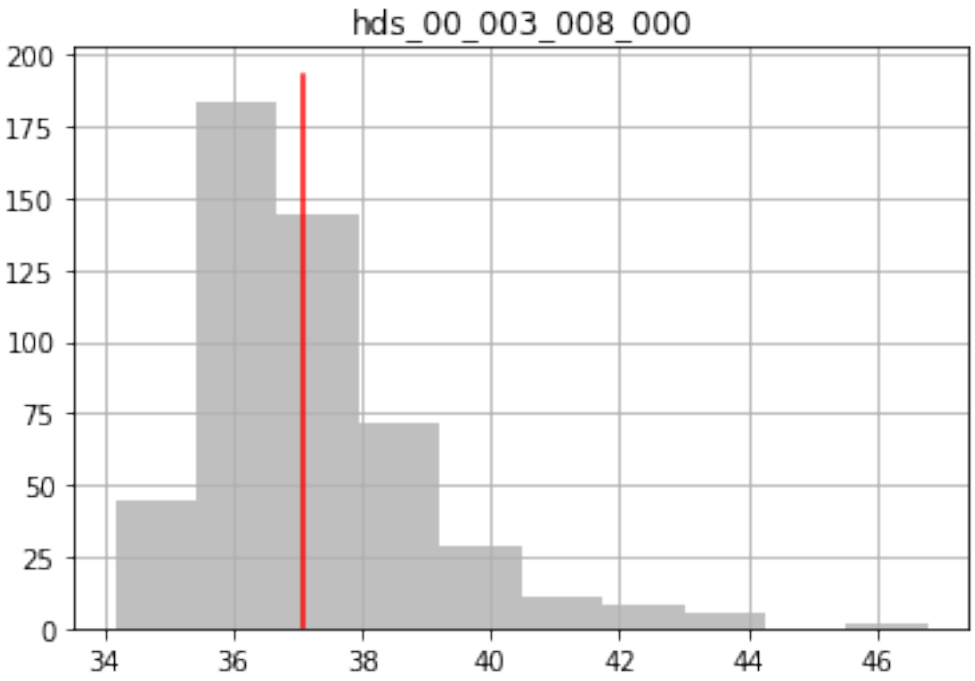
observations:

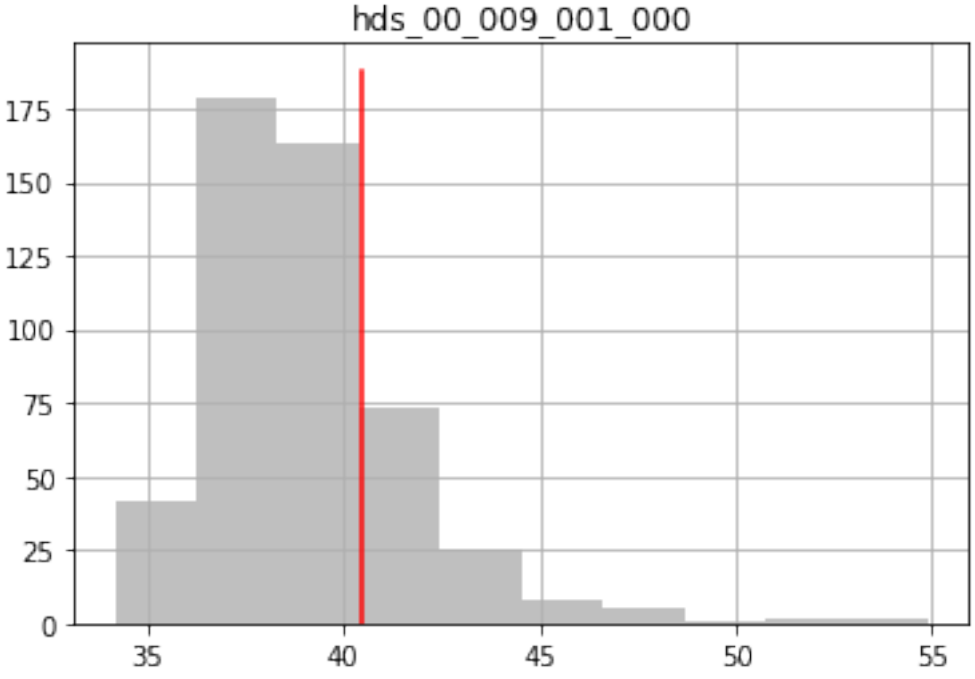
```
In [31]: for oname in pst.nnz_obs_names:
             ax = plt.subplot(111)
             obs_df.loc[:,oname].hist(ax=ax,color="0.5",alpha=0.5)
             ax.plot([obs.loc[oname,"obsval"],obs.loc[oname,"obsval"]],ax.get_ylim(),"r")
             ax.set_title(oname)
             plt.show()
```

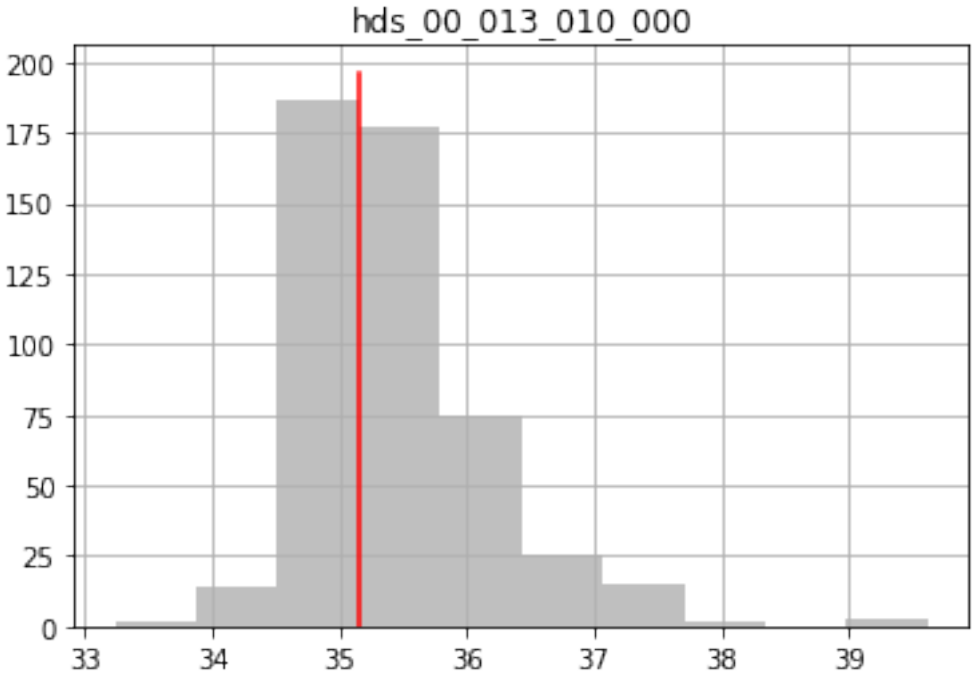


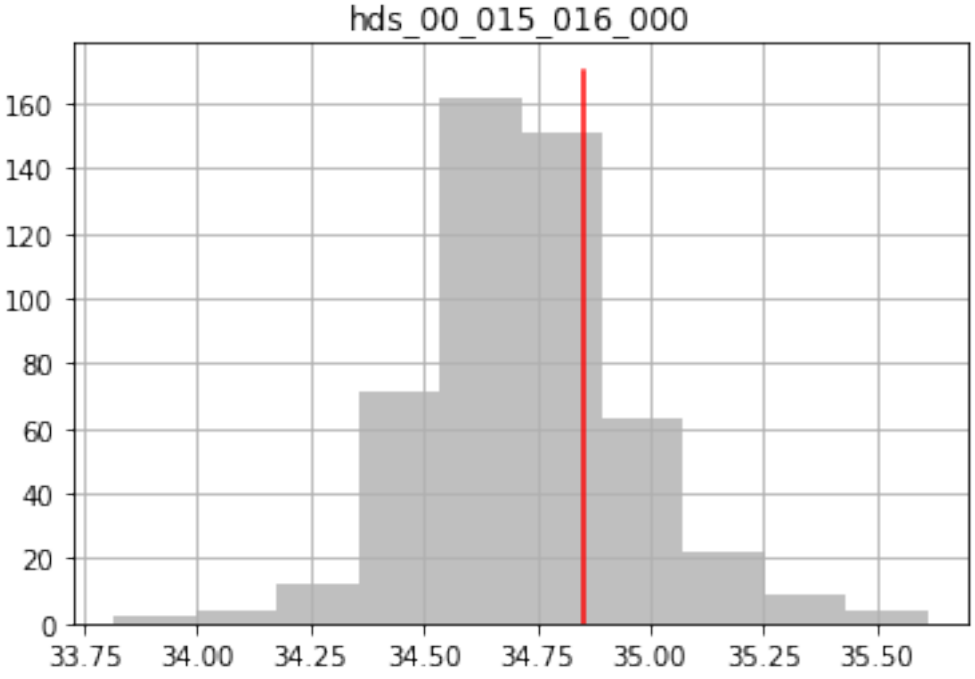

hds_00_002_009_000
200
175
150
125
100
75
50
25
0
34
36
38
40
42
44
46

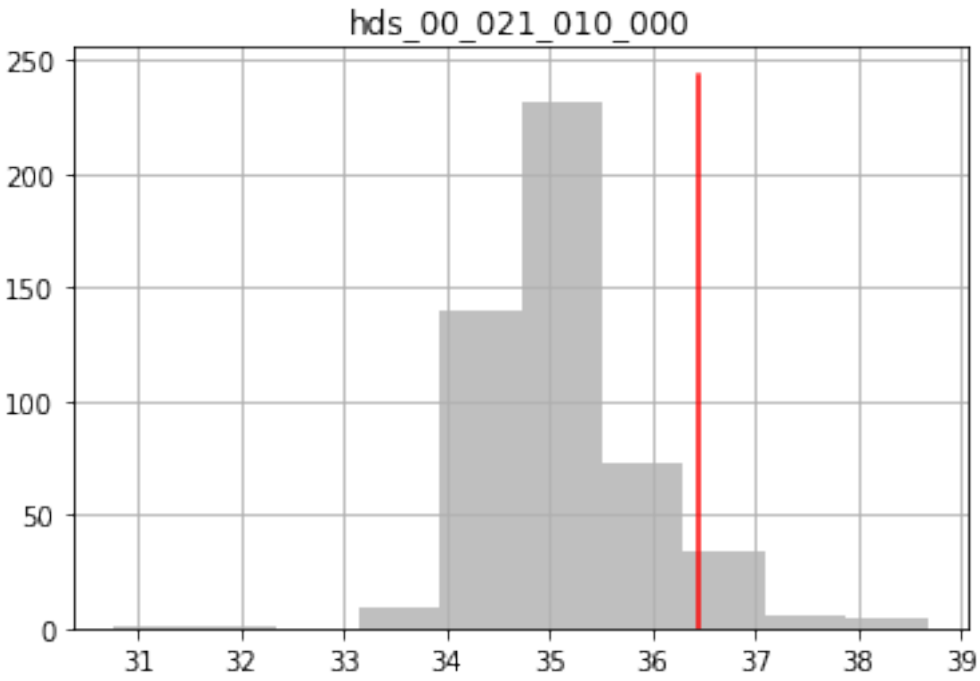

hds_00_002_015_000
140
120
100
80
60
40
20
0
34.25
34.50
34.75
35.00
35.25
35.50
35.75

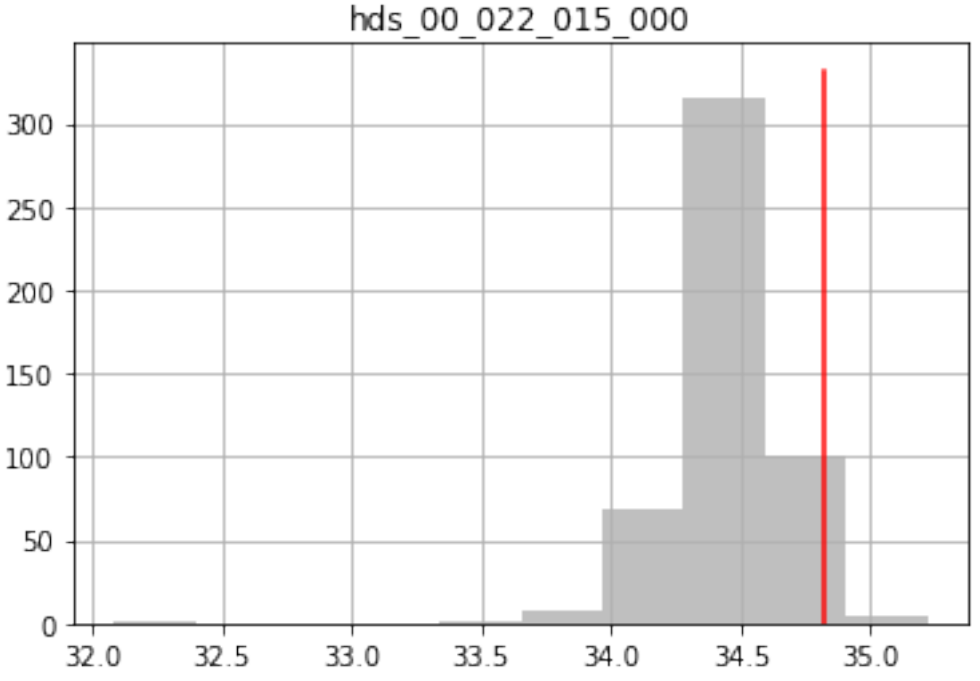

hds_00_003_008_000
200
175
150
125
100
75
50
25
0
34
36
38
40
42
44
46

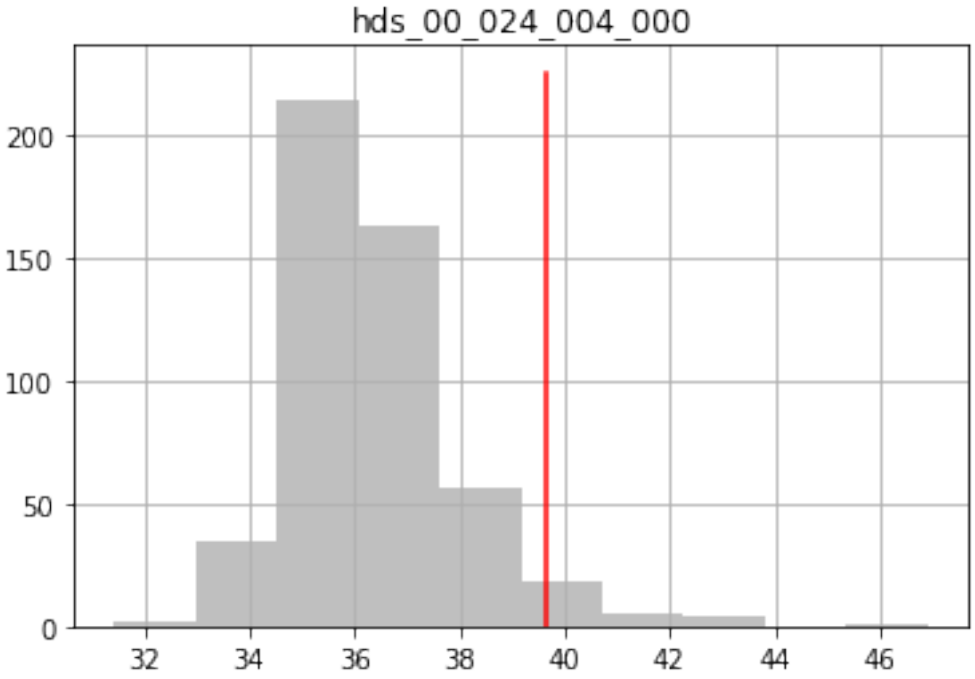

hds_00_009_001_000
175
150
125
100
75
50
25
0
35
40
45
50
55

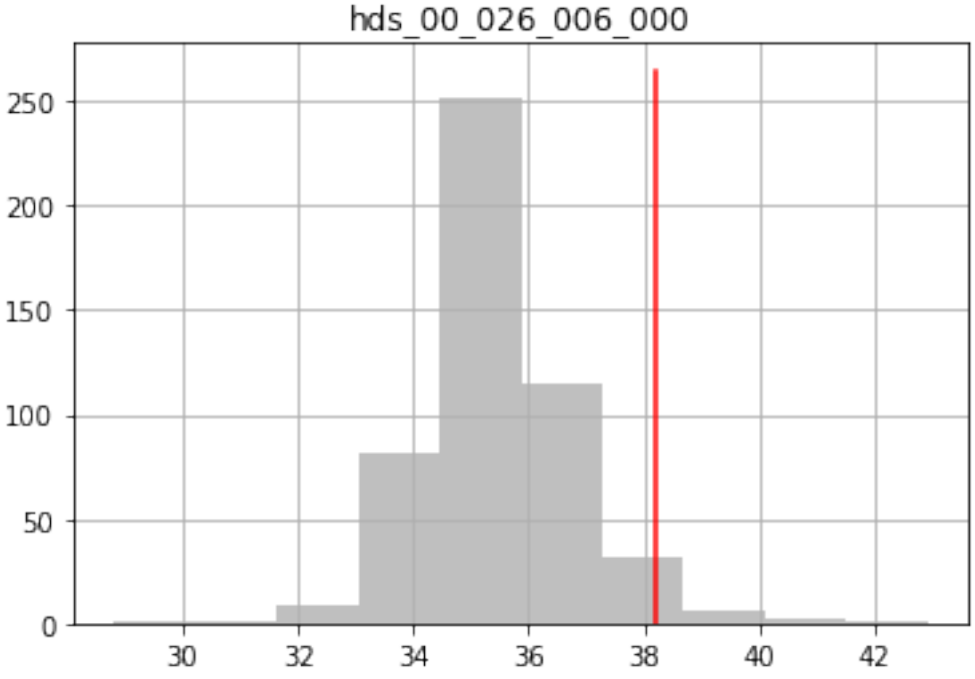

hds_00_013_010_000
200
175
150
125
100
75
50
25
0
33
34
35
36
37
38
39

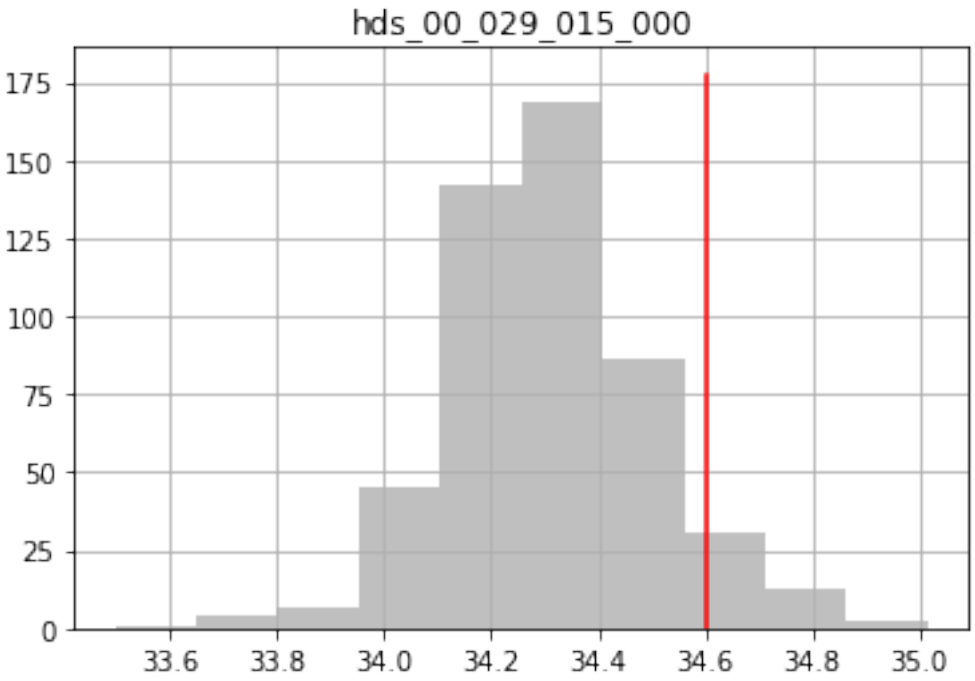

hds_00_015_016_000
160
140
120
100
80
60
40
20
0
33.75
34.00
34.25
34.50
34.75
35.00
35.25
35.50

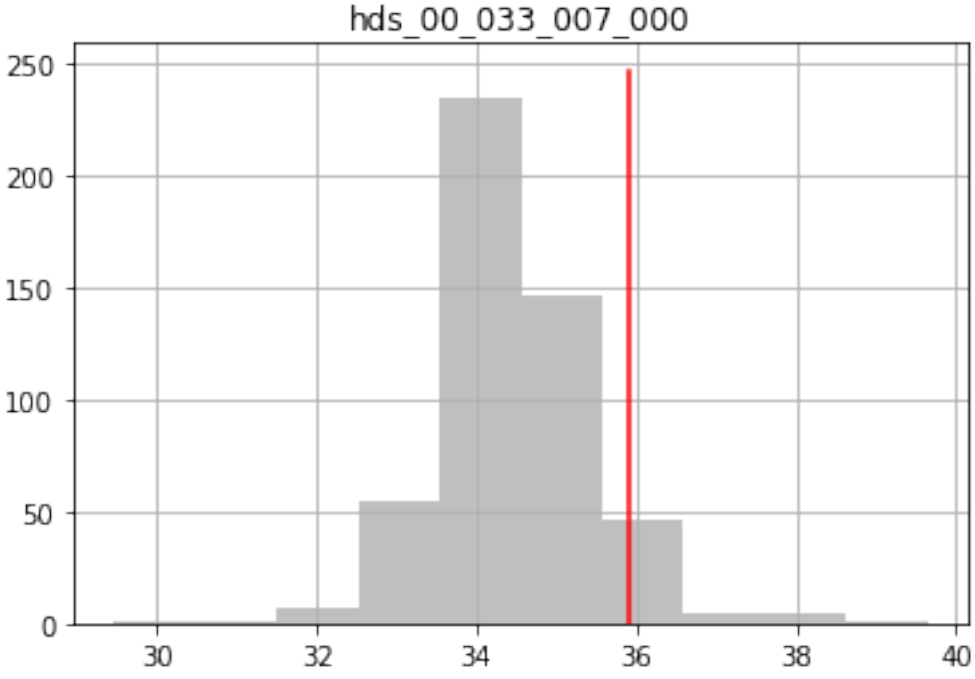

hds_00_021_010_000
250
200
150
100
50
0
31
32
33
34
35
36
37
38
39

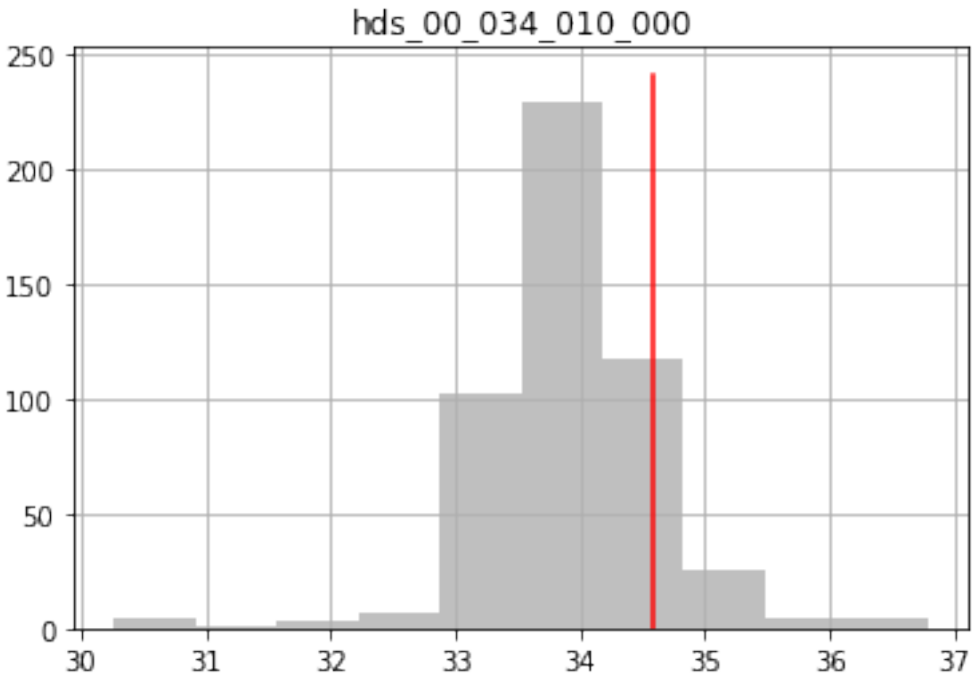

hds_00_022_015_000
300
250
200
150
100
50
0
32.0
32.5
33.0
33.5
34.0
34.5
35.0


hds_00_024_004_000
200
150
100
50
0
32
34
36
38
40
42
44
46


hds_00_026_006_000
250
200
150
100
50
0
30
32
34
36
38
40
42


hds_00_029_015_000
175
150
125
100
75
50
25
0
33.6
33.8
34.0
34.2
34.4
34.6
34.8
35.0


hds_00_033_007_000
250
200
150
100
50
0
30
32
34
36
38
40


hds_00_034_010_000
250
200
150
100
50
0
30
31
32
33
34
35
36
37