prior_montecarlo

May 18, 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
```

1.1 SUPER IMPORTANT: SET HOW MANY PARALLEL WORKERS TO USE

```
In [2]: num_workers = 20
```

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

```
In [3]: t_d = "template"
        pst = pyemu.Pst(os.path.join(t_d,"freyberg.pst"))
```

1.1.2 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 [4]: 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 [5]: 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 [6]: pe.loc[:,"hk031"]
```

```
Out[6]: 0      0.935197
        1      2.122076
        2      1.657241
        3      0.626902
        4      0.780509
        5      0.500825
        6      2.419110
        7      0.558504
        8      0.257237
        9      3.825338
        10     1.413887
        11     0.991652
        12     0.393278
        13     1.427654
        14     4.727755
        15     0.431509
        16     1.441612
        17     3.049608
        18     1.008735
        19     0.415130
        20     3.075104
        21     0.634092
        22     0.552351
        23     0.720671
        24     0.561430
        25     1.205299
        26     0.333346
        27     0.803041
        28     2.133155
        29     2.505986
                 ...
        970    0.434642
        971    0.526258
        972    1.583272
        973    0.277955
        974    0.284029
```

```
975    0.830765
976    0.872251
977    0.714677
978    1.159863
979    1.512948
980    3.750909
981    0.564394
982    0.779535
983    0.176048
984    0.948359
985    0.807764
986    2.040333
987    0.732289
988    0.372329
989    1.369781
990    0.527716
991    1.513590
992    1.304466
993    0.248067
994    3.707392
995    1.205019
996    0.611359
997    3.635739
998    0.593297
999    1.052875
Name: hk031, Length: 1000, dtype: float64
```

In [7]:
```
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[7]:
```
                                    name        group      measured  \
name
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

                          modelled      residual  weight
name
fa_0_19791230         -1.178400e+02  4.876100e+01     0.0
fa_0_19801229         -6.799900e+01 -9.590000e-01     0.0
fa_10_19791230        -4.867000e+01  1.240400e+01     0.0
fa_10_19801229        -5.177400e+00 -3.102560e+01     0.0
fa_11_19791230        -1.340400e+01 -2.396700e+01     0.0
fa_11_19801229        -7.796600e-01 -3.653634e+01     0.0
fa_12_19791230        -3.278900e+01 -7.670000e+00     0.0
fa_12_19801229        -7.499300e-01 -3.966107e+01     0.0
fa_13_19791230        -6.120300e+00 -3.696170e+01     0.0
fa_13_19801229         1.903100e-01 -4.322931e+01     0.0
fa_14_19791230        -8.848300e+01  4.376600e+01     0.0
fa_14_19801229         1.126300e+01 -5.594100e+01     0.0
fa_15_19791230        -4.268900e+00 -4.096410e+01     0.0
fa_15_19801229         1.116600e+00 -4.631460e+01     0.0
fa_16_19791230        -2.666400e+01 -1.832500e+01     0.0
fa_16_19801229         1.543500e+01 -6.039200e+01     0.0
fa_17_19791230        -5.635500e+00 -3.803850e+01     0.0
fa_17_19801229         7.867800e+00 -5.150980e+01     0.0
fa_18_19791230        -5.725000e+00 -3.522800e+01     0.0
fa_18_19801229         2.293400e+01 -6.385600e+01     0.0
fa_19_19791230        -2.970300e-02 -3.615230e+01     0.0
fa_19_19801229         3.335300e+01 -6.950400e+01     0.0
fa_1_19791230         -9.470500e+00 -5.997150e+01     0.0
fa_1_19801229         -5.464700e+00 -6.385730e+01     0.0
fa_20_19791230         2.533400e+01 -5.542000e+01     0.0
fa_20_19801229         1.739600e+02 -2.040150e+02     0.0
fa_21_19791230         3.550900e-01 -3.583909e+01     0.0
fa_21_19801229         2.653500e+00 -3.810550e+01     0.0
fa_22_19791230         1.082900e+01 -5.018100e+01     0.0
fa_22_19801229         8.561700e+01 -1.249350e+02     0.0
...                             ...           ...     ...
hds_02_039_010_000     3.263759e+01 -7.713318e-02     0.0
hds_02_039_010_001     3.260130e+01 -4.086304e-02     0.0
hds_02_039_011_000     3.262994e+01 -6.852341e-02     0.0
hds_02_039_011_001     3.259931e+01 -3.791809e-02     0.0
hds_02_039_012_000     3.267067e+01 -1.050873e-01     0.0
hds_02_039_012_001     3.263583e+01 -7.027054e-02     0.0
hds_02_039_013_000     3.269498e+01 -1.178703e-01     0.0
hds_02_039_013_001     3.266051e+01 -8.341598e-02     0.0
hds_02_039_014_000     3.274148e+01 -1.436729e-01     0.0
hds_02_039_014_001     3.270523e+01 -1.074409e-01     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  -4.305028e+06  1.664892e+06     0.0
vol_drains_19801229  -4.626494e+06  1.722452e+06     0.0
vol_in-out_19791230  -8.276600e+04  8.281100e+04     0.0
vol_in-out_19801229  -8.255600e+04  8.261900e+04     0.0
vol_percent_19791230 -6.000000e-01  6.000000e-01     0.0
vol_percent_19801229 -5.500000e-01  5.500000e-01     0.0
vol_recharg_19791230  1.171976e+07 -6.033230e+05     0.0
vol_recharg_19801229  1.225148e+07 -2.340100e+04     0.0
vol_storage_19791230  6.845375e+05 -6.552992e+05     0.0
vol_storage_19801229  9.301668e+05 -8.988212e+05     0.0
vol_stream__19791230 -1.352142e+06 -3.868352e+06     0.0
vol_stream__19801229 -1.008945e+06 -4.732879e+06     0.0
vol_total_19791230   -8.276600e+04  8.281100e+04     0.0
vol_total_19801229   -8.255600e+04  8.261900e+04     0.0
vol_wells_19791230   -6.829895e+06  3.544895e+06     0.0
vol_wells_19801229   -7.628768e+06  4.015268e+06     0.0
part_status           2.000000e+00  1.000000e+10     0.0
part_time             6.457774e+02  9.999999e+09     0.0

[4436 rows x 6 columns]
```

1.1.3 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 [8]: m_d = "master_prior_sweep"
        pyemu.os_utils.start_slaves(t_d,"pestpp-swp","freyberg.pst",num_slaves=num_workers,sla
```

1.1.4 Load the output ensemble and plot a few things

```
In [9]: 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: 1000
```

drop any failed runs

```
In [10]: 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: 1000
```

```
In [11]: obs_df.iloc[0,:]
```

Out[11]:
```
input_run_id            0.000000e+00
failed_flag             0.000000e+00
phi                     1.887895e+06
meas_phi                1.887895e+06
regul_phi               0.000000e+00
vol_wells               0.000000e+00
vol_recharg             0.000000e+00
flx_recharg             0.000000e+00
obgnme                  0.000000e+00
vol_constan             0.000000e+00
flx_total               0.000000e+00
flx_percent             0.000000e+00
flx_storage             0.000000e+00
flx_wells               0.000000e+00
vol_percent             0.000000e+00
vol_stream_             0.000000e+00
vol_drains              0.000000e+00
flx_in-out              0.000000e+00
vol_storage             0.000000e+00
hds                     0.000000e+00
vol_total               0.000000e+00
flx_drains              0.000000e+00
flaqx                   0.000000e+00
vol_in-out              0.000000e+00
flout                   0.000000e+00
calhead                 1.875375e+01
flx_stream_             0.000000e+00
flx_constan             0.000000e+00
calflux                 1.887876e+06
fa_0_19791230          -1.178400e+02
                            ...
hds_02_039_010_000      3.263759e+01
hds_02_039_010_001      3.260130e+01
hds_02_039_011_000      3.262994e+01
hds_02_039_011_001      3.259931e+01
hds_02_039_012_000      3.267067e+01
hds_02_039_012_001      3.263583e+01
hds_02_039_013_000      3.269498e+01
hds_02_039_013_001      3.266051e+01
hds_02_039_014_000      3.274148e+01
hds_02_039_014_001      3.270523e+01
vol_constan_19791230    0.000000e+00
vol_constan_19801229    0.000000e+00
vol_drains_19791230    -4.305028e+06
vol_drains_19801229    -4.626494e+06
vol_in-out_19791230    -8.276600e+04
vol_in-out_19801229    -8.255600e+04
vol_percent_19791230   -6.000000e-01
```

```
vol_percent_19801229   -5.500000e-01
vol_recharg_19791230    1.171976e+07
vol_recharg_19801229    1.225148e+07
vol_storage_19791230    6.845375e+05
vol_storage_19801229    9.301668e+05
vol_stream__19791230   -1.352142e+06
vol_stream__19801229   -1.008945e+06
vol_total_19791230     -8.276600e+04
vol_total_19801229     -8.255600e+04
vol_wells_19791230     -6.829895e+06
vol_wells_19801229     -7.628768e+06
part_status             2.000000e+00
part_time               6.457774e+02
Name: 0, Length: 4465, dtype: float64
```

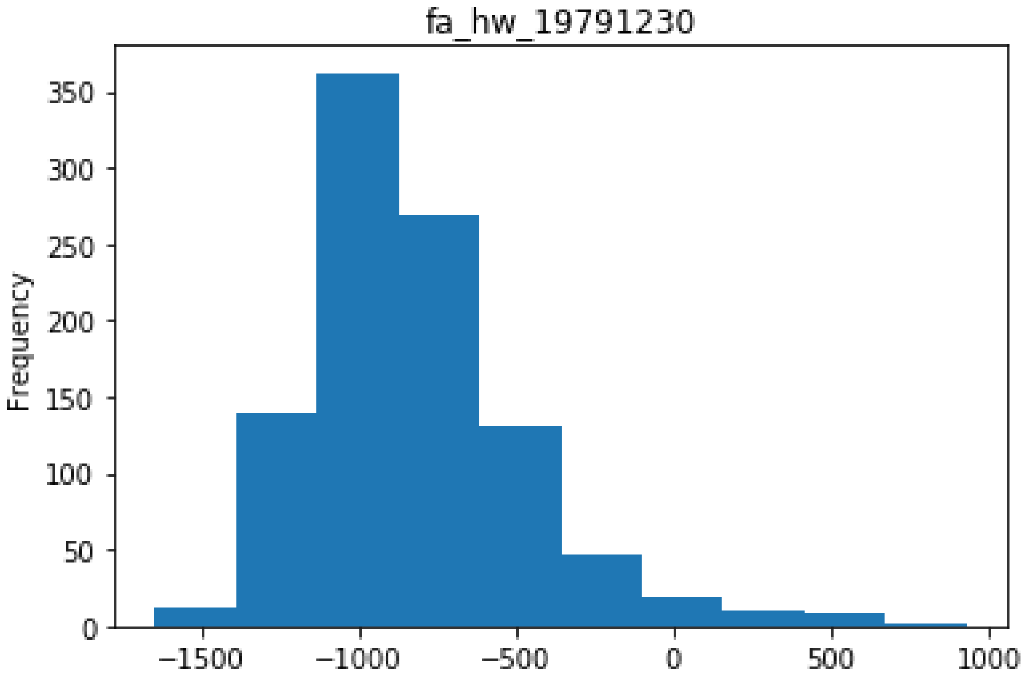
1.1.5 confirm which quantities were identified as forecasts

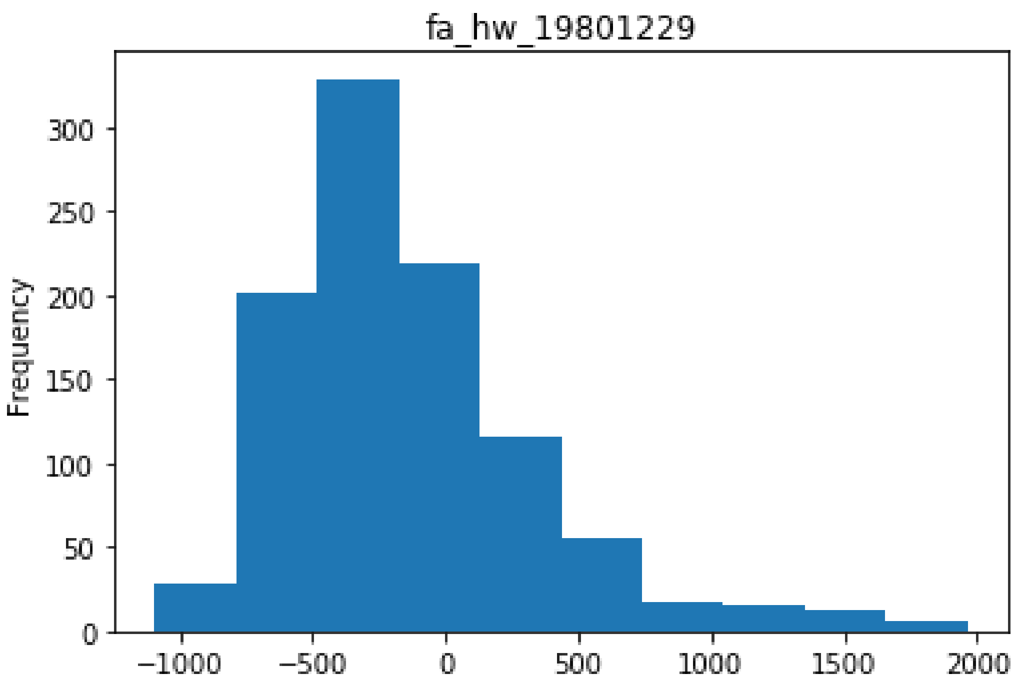
```
In [12]: fnames = pst.pestpp_options["forecasts"].split(',')
         fnames
```

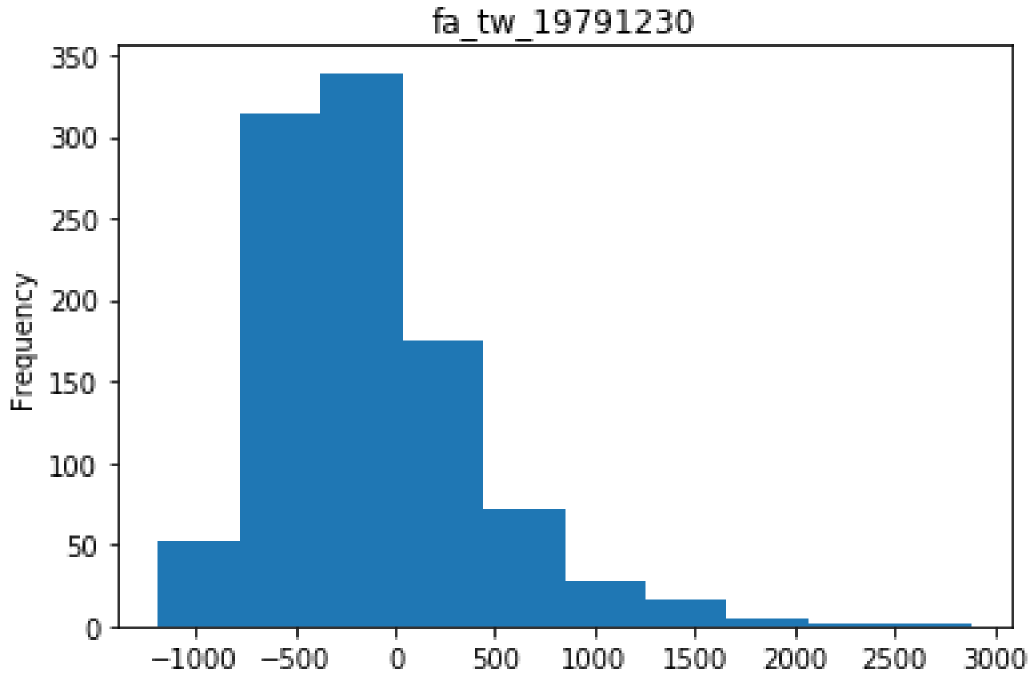
```
Out[12]: ['fa_hw_19791230',
          'fa_hw_19801229',
          'fa_tw_19791230',
          'fa_tw_19801229',
          'hds_00_013_002_000',
          'hds_00_013_002_001',
          'part_time',
          'part_status']
```

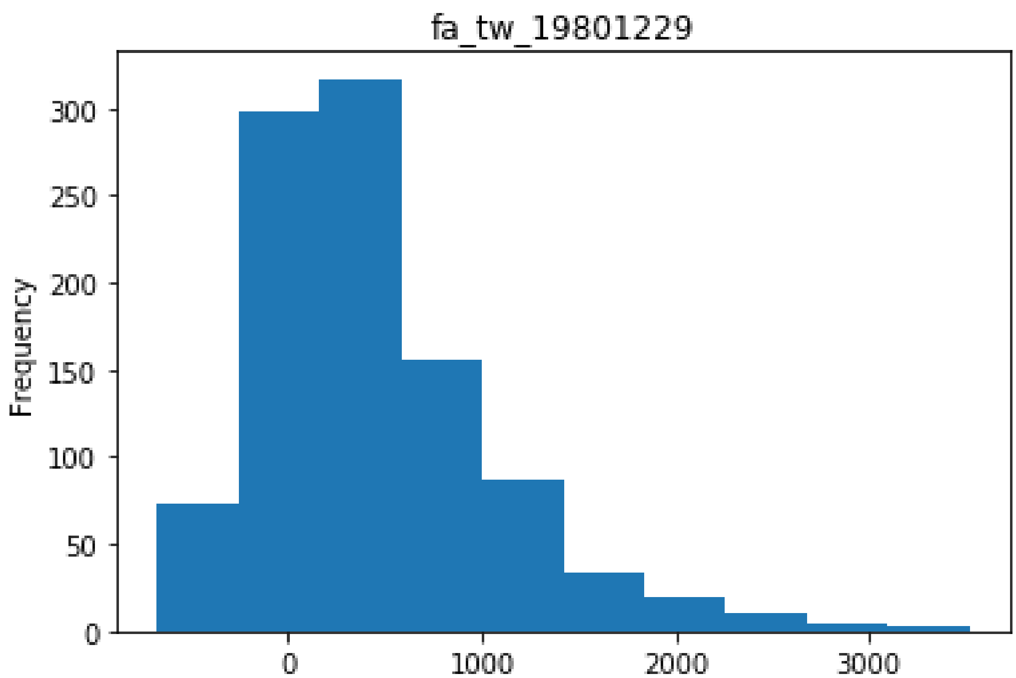
1.1.6 now we can plot the distributions of each forecast

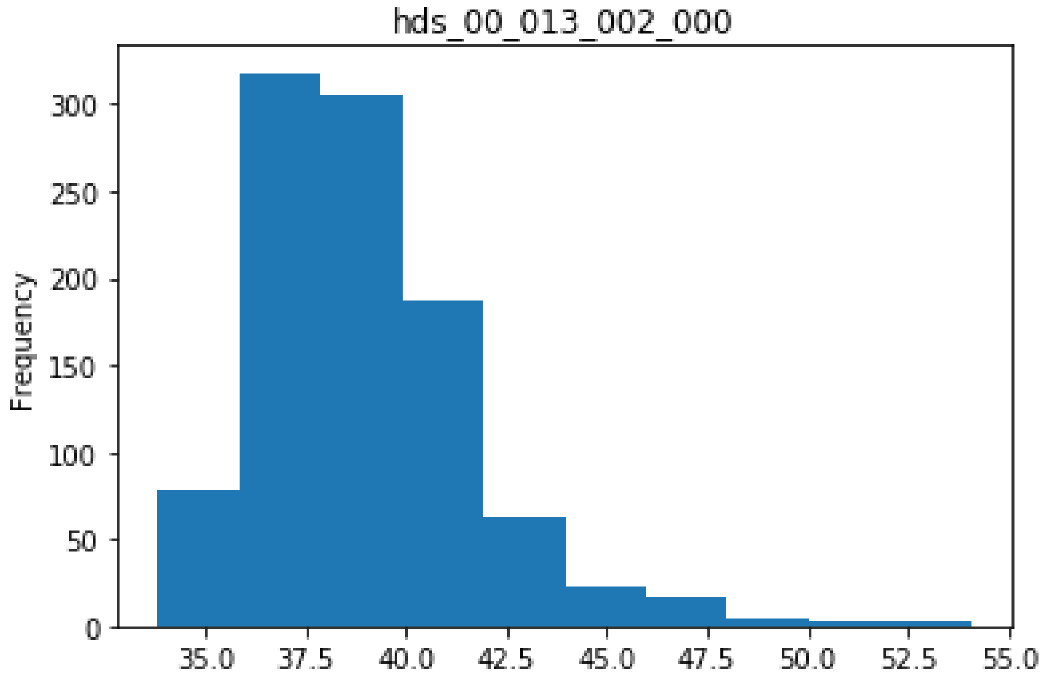
```
In [13]: for forecast in fnames:
             plt.figure()
             ax = obs_df.loc[:,forecast].plot(kind="hist")
             ax.set_title(forecast)
         plt.show()
```

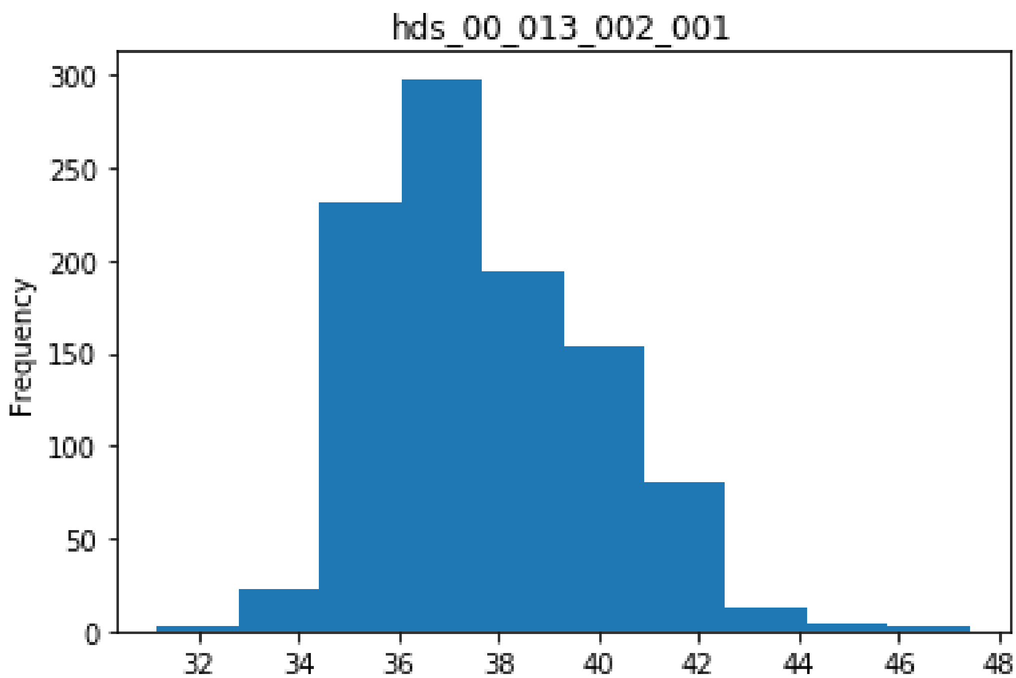

fa_hw_19791230
Frequency
0
50
100
150
200
250
300
350
−1500
−1000
−500
0
500
1000

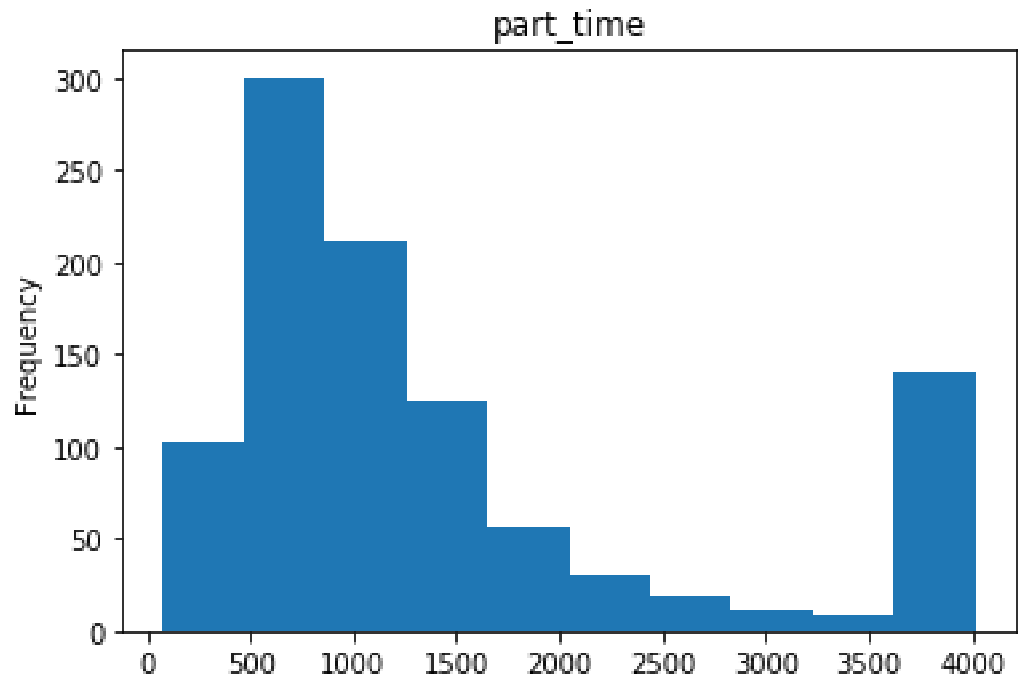

fa_hw_19801229
Frequency
0
50
100
150
200
250
300
−1000
−500
0
500
1000
1500
2000

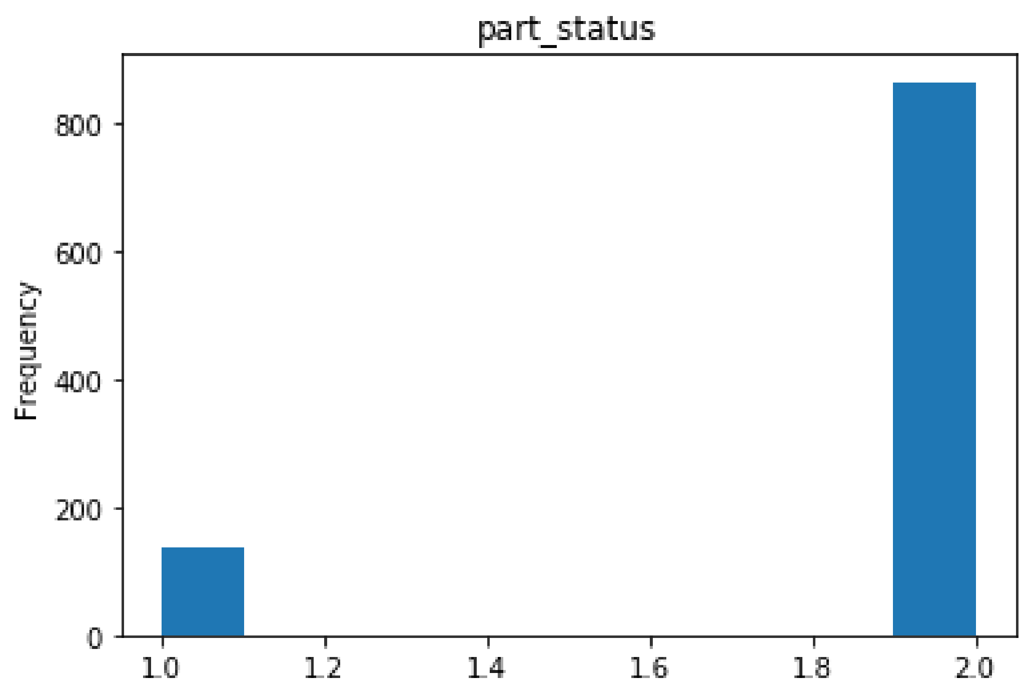

fa_tw_19791230
Frequency
350
300
250
200
150
100
50
0
−1000
−500
0
500
1000
1500
2000
2500
3000


fa_tw_19801229
Frequency
300
250
200
150
100
50
0
0
1000
2000
3000


hds_00_013_002_000
Frequency
0
50
100
150
200
250
300
35.0
37.5
40.0
42.5
45.0
47.5
50.0
52.5
55.0

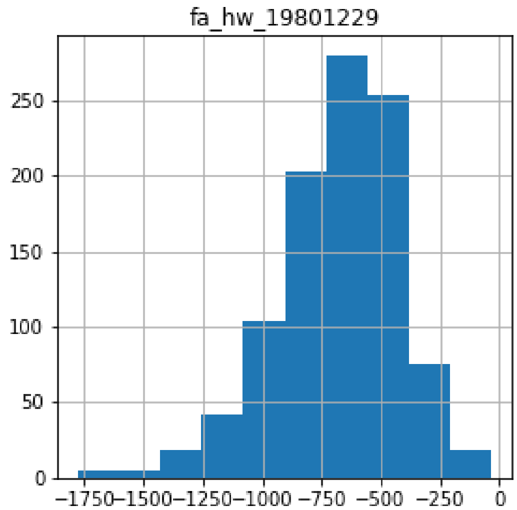

hds_00_013_002_001
Frequency
0
50
100
150
200
250
300
32
34
36
38
40
42
44
46
48

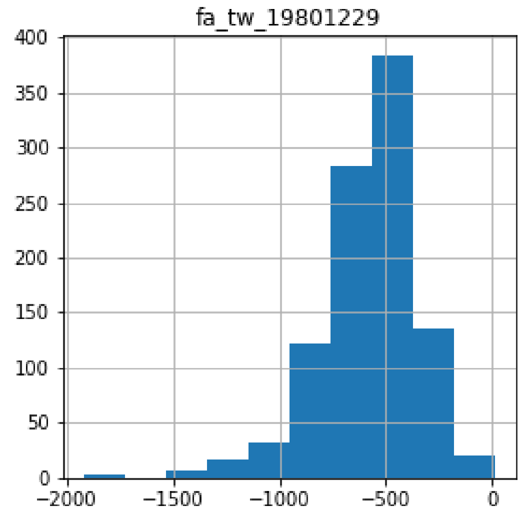


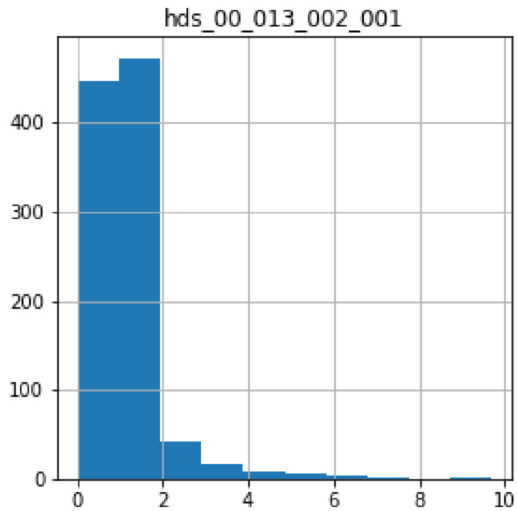


We see that under scenario conditions, many more realizations for the flow to the aquifer in the headwaters are postive (as expected). Lets difference these two:

```
In [14]: 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)

1.1.7 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 [15]:
```
sorted_vals = obs_df.loc[:,"part_time"].sort_values()
idx = sorted_vals.index[100]
idx
sorted_vals
```

Out[15]:
```
run_id
120      76.4341
385     159.3529
100     165.7032
82      170.2694
980     207.3890
71      230.5359
392     235.9687
663     240.3898
217     246.4102
751     247.6348
669     251.1703
596     255.4728
281     260.8926
452     267.2885
632     273.5754
315     275.9690
732     277.0921
954     283.0271
115     284.2840
653     293.4041
457     294.3939
398     302.9174
187     303.7302
723     308.5808
543     308.8079
797     310.4536
43      311.5282
254     311.6480
546     311.9860
761     322.6084
          ...
892    4015.0000
893    4015.0000
896    4015.0000
897    4015.0000
881    4015.0000
95     4015.0000
860    4015.0000
855    4015.0000
```

```
507    4015.0000
819    4015.0000
364    4015.0000
511    4015.0000
655    4015.0000
150    4015.0000
826    4015.0000
361    4015.0000
355    4015.0000
857    4015.0000
353    4015.0000
517    4015.0000
148    4015.0000
648    4015.0000
521    4015.0000
36     4015.0000
35     4015.0000
523    4015.0000
851    4015.0000
337    4015.0000
350    4015.0000
129    4015.0000
Name: part_time, Length: 1000, dtype: float64
```

In [16]:
```
obs_df.loc[idx,pst.nnz_obs_names]
```

Out[16]:
```
fo_39_19791230         10530.000000
hds_00_002_009_000        36.178482
hds_00_002_015_000        34.927410
hds_00_003_008_000        36.352409
hds_00_009_001_000        37.531170
hds_00_013_010_000        34.860771
hds_00_015_016_000        34.580383
hds_00_021_010_000        34.844711
hds_00_022_015_000        34.249882
hds_00_024_004_000        35.689373
hds_00_026_006_000        35.276196
hds_00_029_015_000        34.393169
hds_00_033_007_000        34.580276
hds_00_034_010_000        34.191792
Name: 495, dtype: float64
```

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

In [17]:
```
obs_df.loc[idx,fnames]
```

Out[17]:
```
fa_hw_19791230       -718.895900
fa_hw_19801229        348.943460
fa_tw_19791230        -38.232140
```

```
fa_tw_19801229        789.050200
hds_00_013_002_000     37.380268
hds_00_013_002_001     36.158752
part_time             466.978300
part_status             2.000000
Name: 495, 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 [18]: 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 [19]: 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[19]: 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

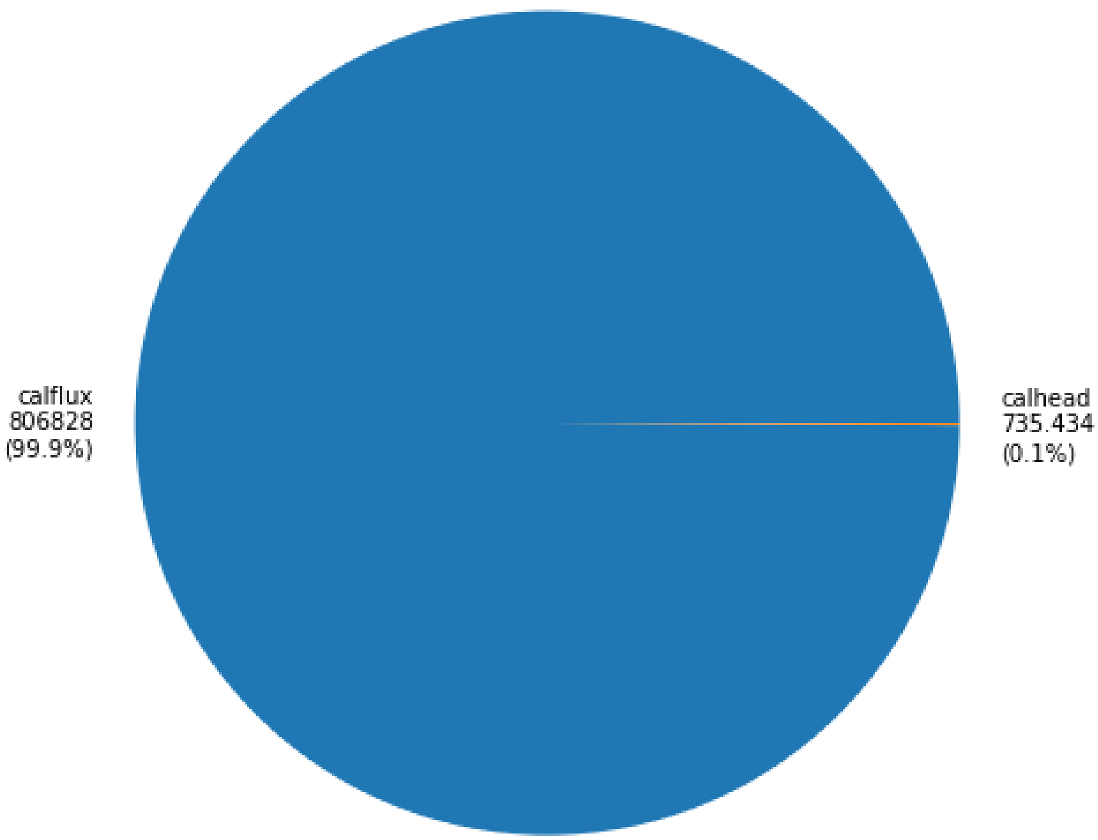
```
In [20]: pst.write(os.path.join(t_d,"freyberg.pst"))
         pyemu.os_utils.run("pestpp-ies freyberg.pst",cwd=t_d)
```

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

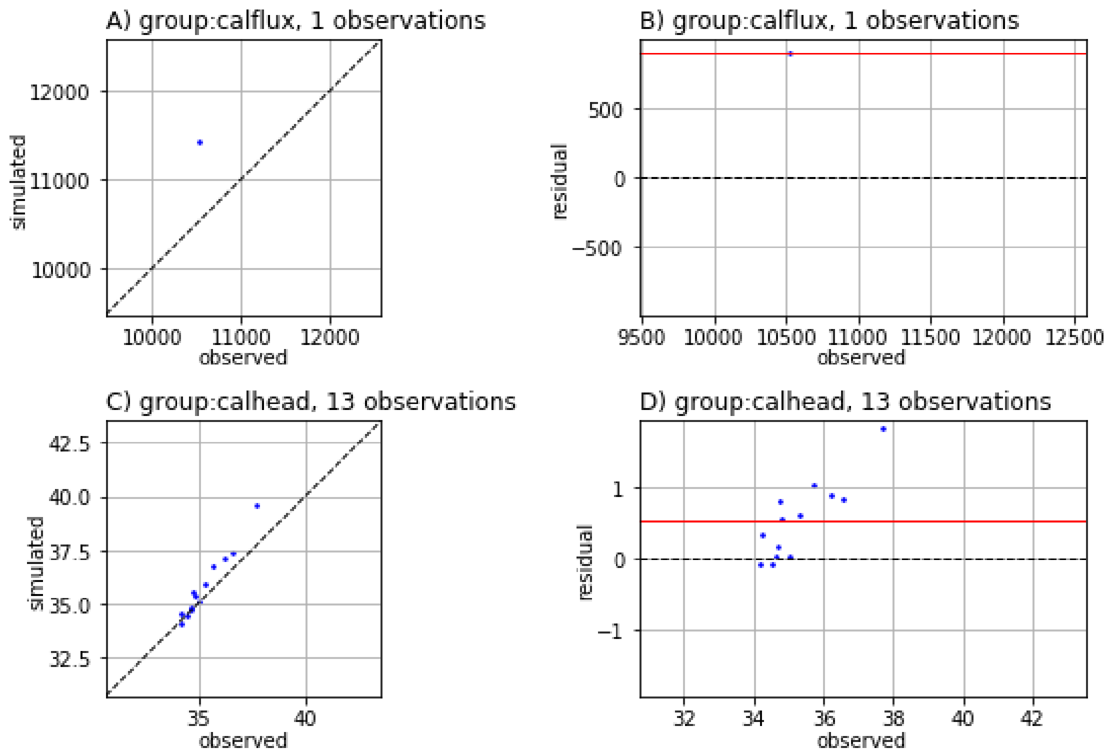
Now we can read in the results and make some figures showing residuals and the balance of the objective function

```
In [21]: 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,:]
```

```
807563.2527031198
Here are the non-zero weighted observation names


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




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



Out[21]:

```
                                    name    group      measured      modelled  \
name
fo_39_19791230            fo_39_19791230  calflux  10531.764052  11430.000000
```

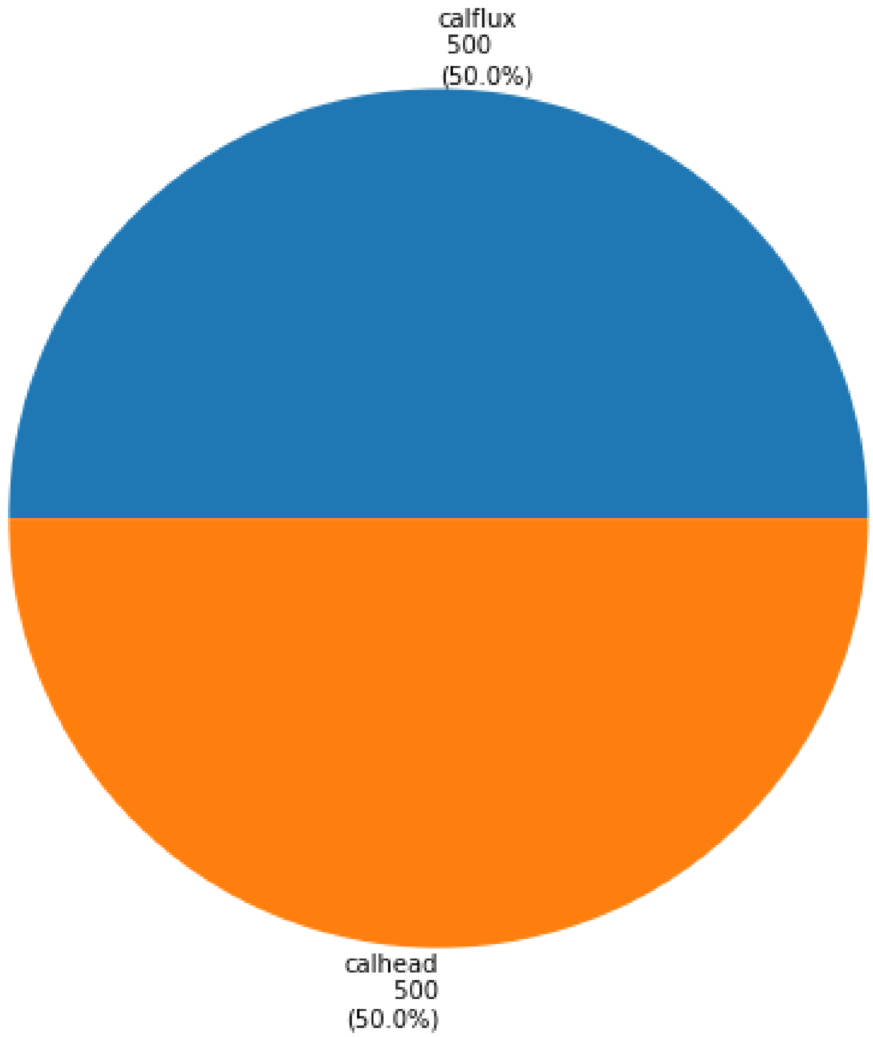
```
hds_00_002_009_000  hds_00_002_009_000  calhead     36.218498     37.107498
hds_00_002_015_000  hds_00_002_015_000  calhead     35.025284     35.045185
hds_00_003_008_000  hds_00_003_008_000  calhead     36.576499     37.397289
hds_00_009_001_000  hds_00_009_001_000  calhead     37.717926     39.546417
hds_00_013_010_000  hds_00_013_010_000  calhead     34.763043     35.571774
hds_00_015_016_000  hds_00_015_016_000  calhead     34.675392     34.835716
hds_00_021_010_000  hds_00_021_010_000  calhead     34.829576     35.386250
hds_00_022_015_000  hds_00_022_015_000  calhead     34.239560     34.577492
hds_00_024_004_000  hds_00_024_004_000  calhead     35.730433     36.760464
hds_00_026_006_000  hds_00_026_006_000  calhead     35.290600     35.896149
hds_00_029_015_000  hds_00_029_015_000  calhead     34.538597     34.453842
hds_00_033_007_000  hds_00_033_007_000  calhead     34.656380     34.678810
hds_00_034_010_000  hds_00_034_010_000  calhead     34.203959     34.118073

                      residual  weight
name
fo_39_19791230     -898.235948     1.0
hds_00_002_009_000   -0.889000    10.0
hds_00_002_015_000   -0.019901    10.0
hds_00_003_008_000   -0.820791    10.0
hds_00_009_001_000   -1.828492    10.0
hds_00_013_010_000   -0.808730    10.0
hds_00_015_016_000   -0.160324    10.0
hds_00_021_010_000   -0.556674    10.0
hds_00_022_015_000   -0.337932    10.0
hds_00_024_004_000   -1.030031    10.0
hds_00_026_006_000   -0.605549    10.0
hds_00_029_015_000    0.084755    10.0
hds_00_033_007_000   -0.022430    10.0
hds_00_034_010_000    0.085887    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 [22]:
```
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[22]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x108088b70>
```



Lets see what the new flux observation weight is:

```
In [23]: pst.observation_data.loc[pst.nnz_obs_names,"weight"]
```

```
Out[23]: obsnme
         fo_39_19791230         0.024894
         hds_00_002_009_000     8.245425
         hds_00_002_015_000     8.245425
         hds_00_003_008_000     8.245425
         hds_00_009_001_000     8.245425
         hds_00_013_010_000     8.245425
         hds_00_015_016_000     8.245425
         hds_00_021_010_000     8.245425
         hds_00_022_015_000     8.245425
```

```
hds_00_024_004_000    8.245425
hds_00_026_006_000    8.245425
hds_00_029_015_000    8.245425
hds_00_033_007_000    8.245425
hds_00_034_010_000    8.245425
Name: weight, dtype: float64
```

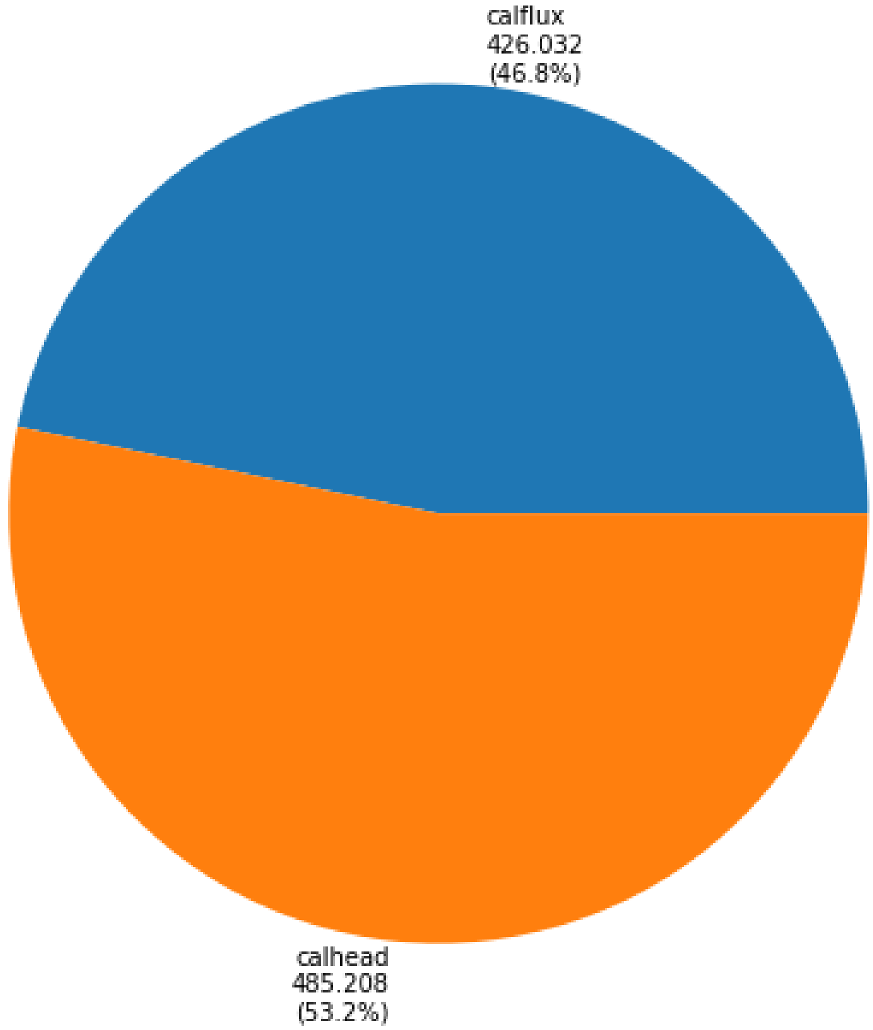
Now, for some super trickery: since we changed the weight, we need to generate the observation noise using these new weights for the error model (so meta!)

In [24]:
```
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[24]:
```
obsnme
fo_39_19791230        70.862570
hds_00_002_009_000     0.048531
hds_00_002_015_000     0.118701
hds_00_003_008_000     0.271774
hds_00_009_001_000     0.226496
hds_00_013_010_000    -0.118524
hds_00_015_016_000     0.115226
hds_00_021_010_000    -0.018357
hds_00_022_015_000    -0.012518
hds_00_024_004_000     0.049797
hds_00_026_006_000     0.017470
hds_00_029_015_000     0.176373
hds_00_033_007_000     0.092298
hds_00_034_010_000     0.014757
Name: weight, dtype: float64
```

In [25]:
```
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
911.2398320951618
```



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 [26]: 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 [27]:
```
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,:]
```

```
17.528847235135252
```

Out[27]:
```
                                 name    group      measured      modelled  \
name
fo_39_19791230         fo_39_19791230  calflux  10600.862570  10530.000000
hds_00_002_009_000  hds_00_002_009_000  calhead     36.227013     36.178482
hds_00_002_015_000  hds_00_002_015_000  calhead     35.046111     34.927410
hds_00_003_008_000  hds_00_003_008_000  calhead     36.624184     36.352409
hds_00_009_001_000  hds_00_009_001_000  calhead     37.757666     37.531170
hds_00_013_010_000  hds_00_013_010_000  calhead     34.742248     34.860771
hds_00_015_016_000  hds_00_015_016_000  calhead     34.695609     34.580383
hds_00_021_010_000  hds_00_021_010_000  calhead     34.826355     34.844711
hds_00_022_015_000  hds_00_022_015_000  calhead     34.237363     34.249882
hds_00_024_004_000  hds_00_024_004_000  calhead     35.739170     35.689373
hds_00_026_006_000  hds_00_026_006_000  calhead     35.293665     35.276196
hds_00_029_015_000  hds_00_029_015_000  calhead     34.569543     34.393169
hds_00_033_007_000  hds_00_033_007_000  calhead     34.672575     34.580276
hds_00_034_010_000  hds_00_034_010_000  calhead     34.206548     34.191792

                     residual    weight
name
fo_39_19791230      70.862570  0.024894
hds_00_002_009_000   0.048531  8.245425
hds_00_002_015_000   0.118701  8.245425
hds_00_003_008_000   0.271774  8.245425
hds_00_009_001_000   0.226496  8.245425
hds_00_013_010_000  -0.118524  8.245425
hds_00_015_016_000   0.115226  8.245425
hds_00_021_010_000  -0.018357  8.245425
hds_00_022_015_000  -0.012518  8.245425
hds_00_024_004_000   0.049797  8.245425
hds_00_026_006_000   0.017470  8.245425
hds_00_029_015_000   0.176373  8.245425
hds_00_033_007_000   0.092298  8.245425
hds_00_034_010_000   0.014757  8.245425
```

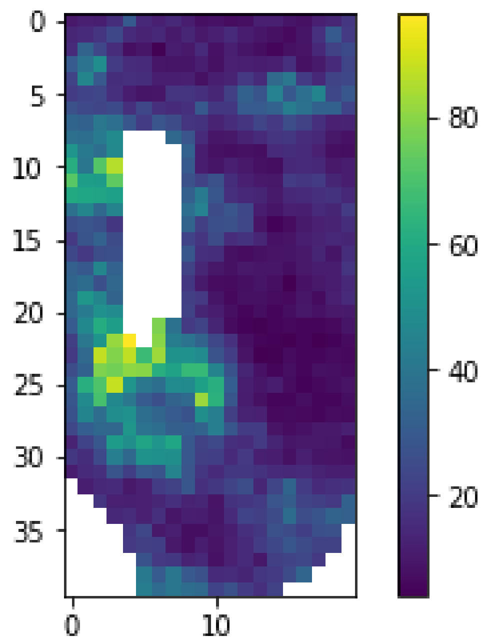
The residual should be exactly the noise values from above. Lets load the model (that was just run using the true pars) and check some things

In [28]:
```
m = flopy.modflow.Modflow.load("freyberg.nam",model_ws=m_d)
```

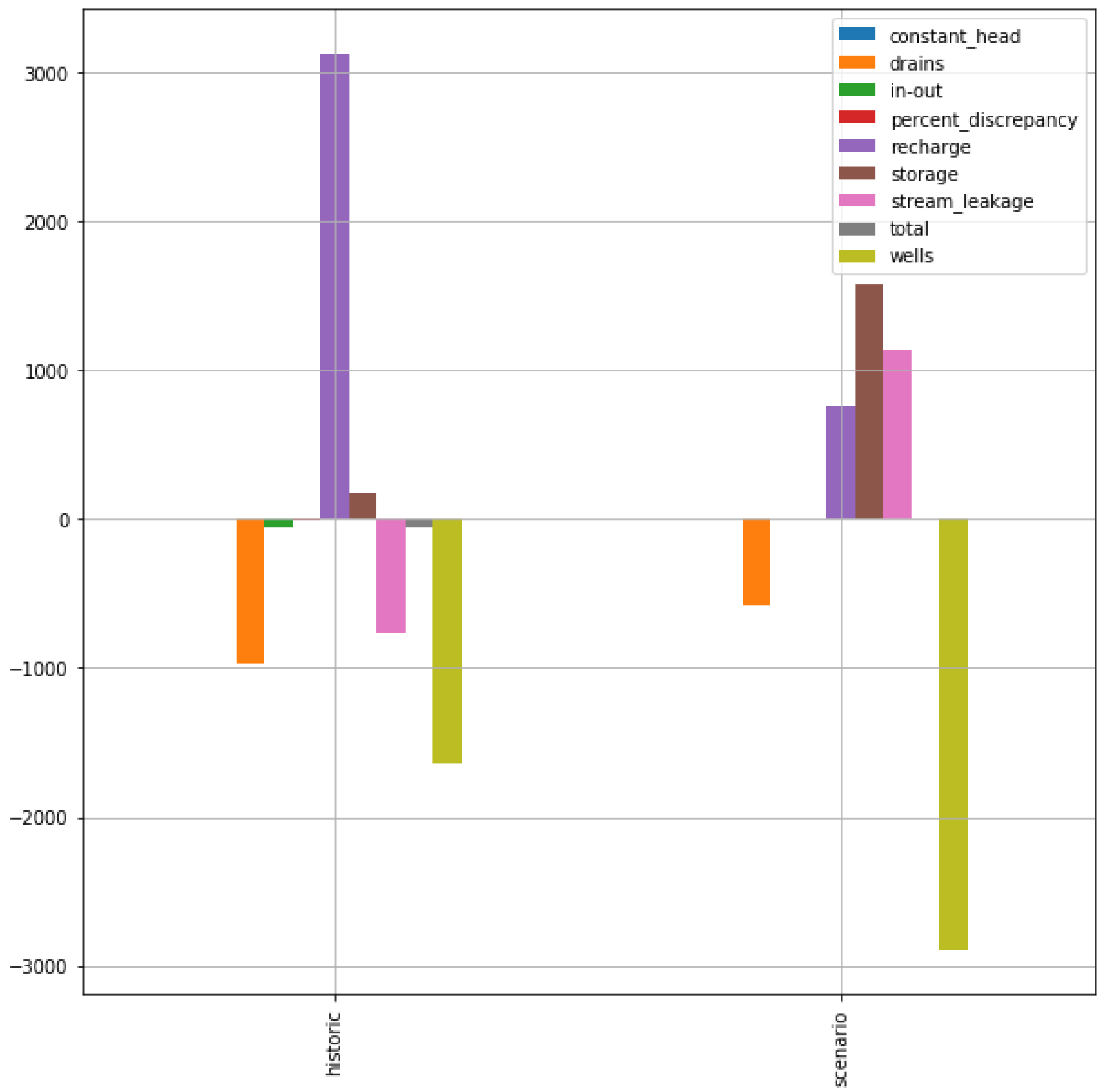
In [29]:
```
a = m.upw.hk[0].array
#a = m.rch.rech[0].array
```

```
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()
```

```
4.05637 96.48033
```



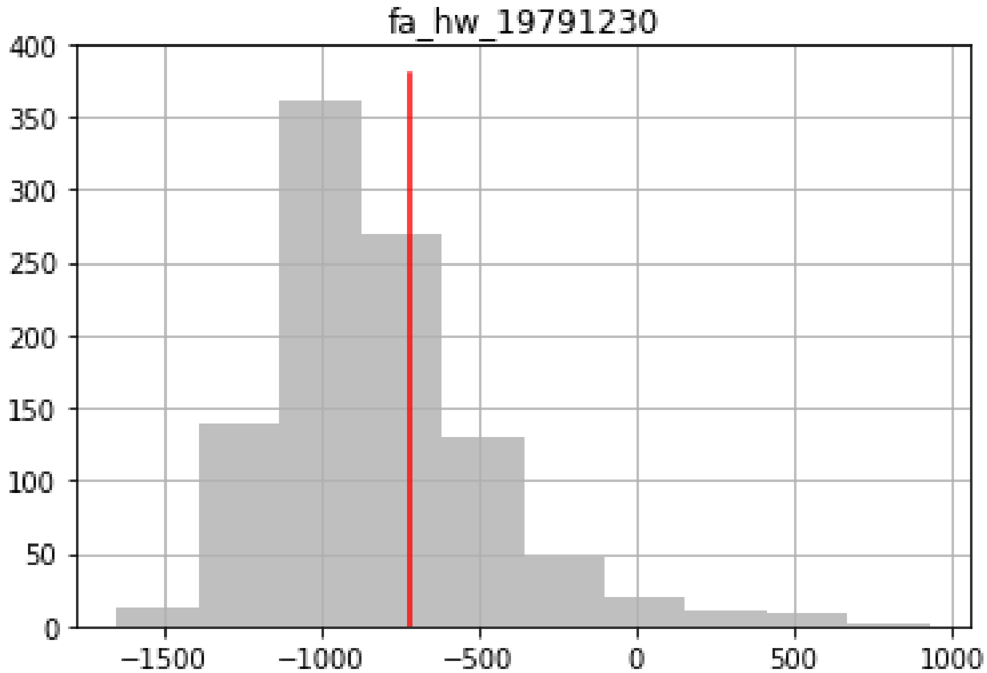

```
In [30]: lst = flopy.utils.MfListBudget(os.path.join(m_d,"freyberg.list"))
         df = lst.get_dataframes(diff=True)[0]
         ax = df.plot(kind="bar",figsize=(10,10), grid=True)
         a = ax.set_xticklabels(["historic","scenario"],rotation=90)
         plt.show(ax)
```

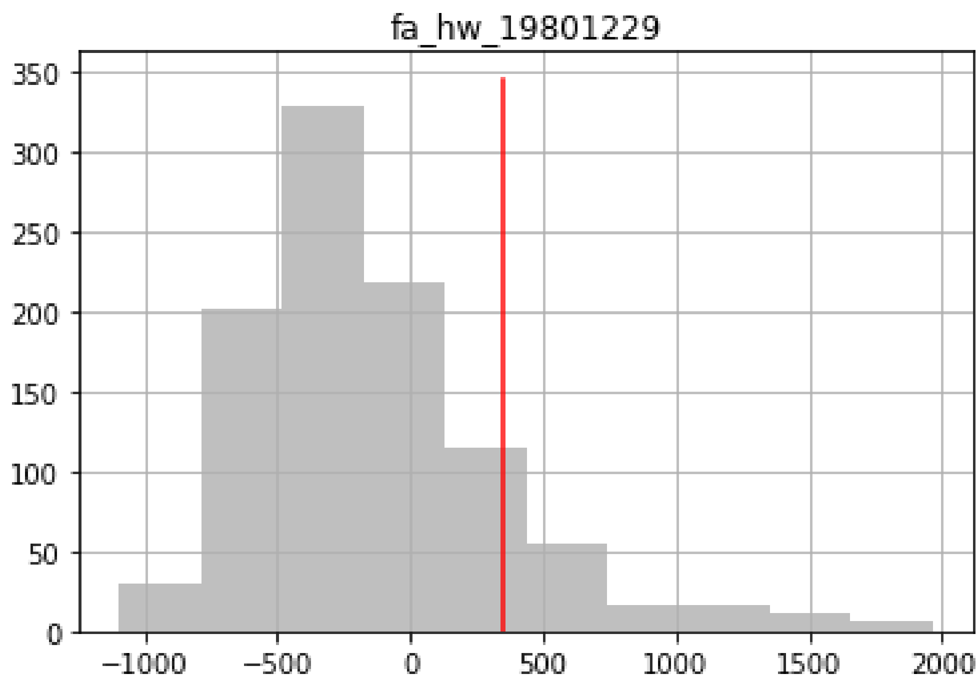


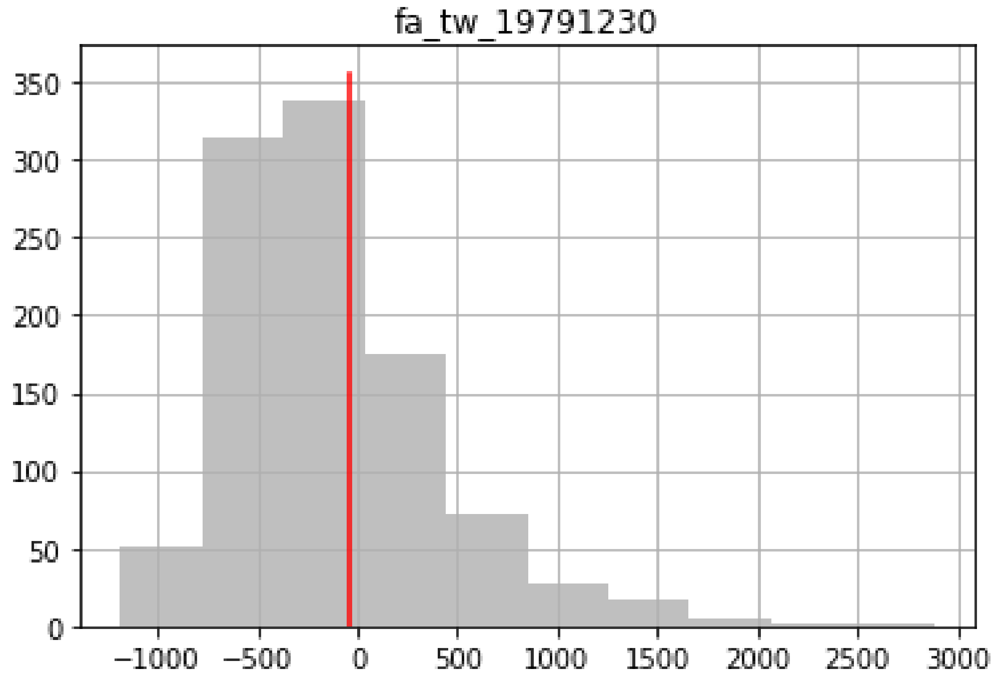
1.1.8 see how our existing observation ensemble compares to the truth

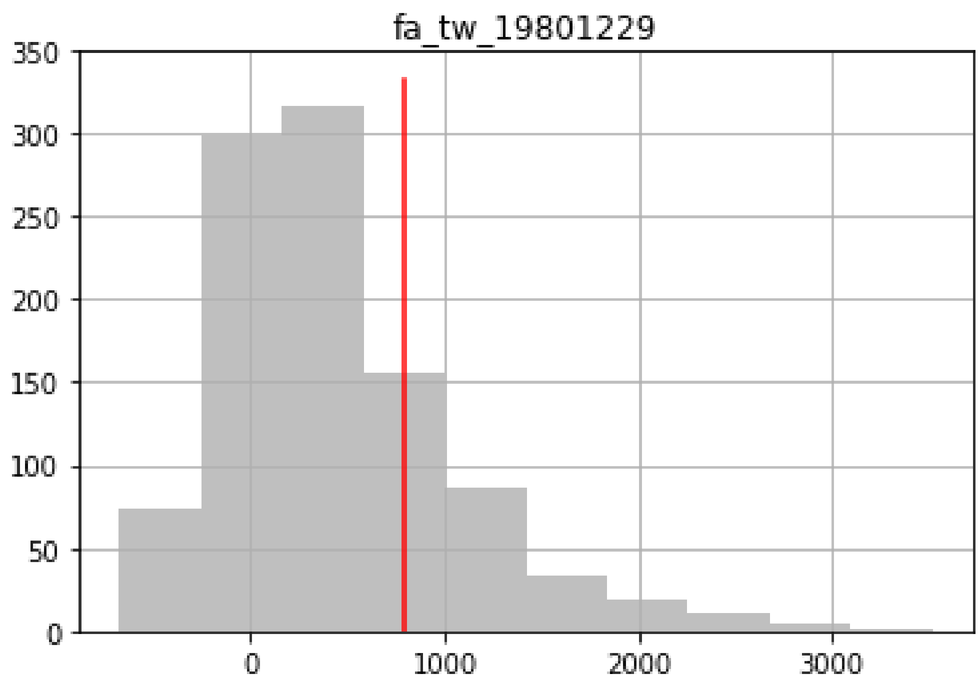
forecasts:

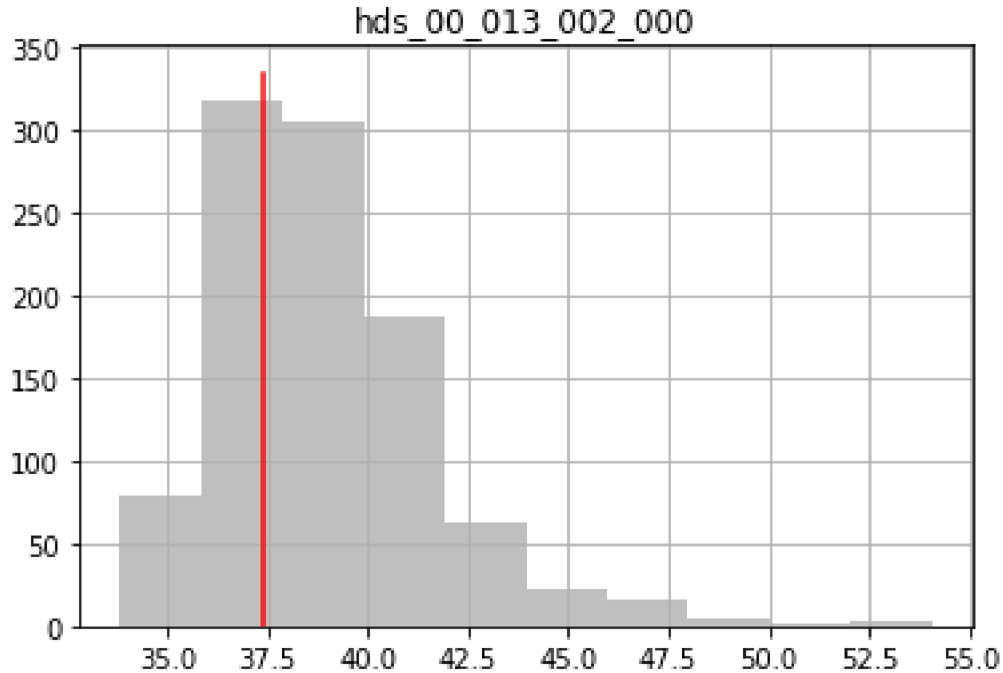
```
In [31]: obs = pst.observation_data
         plt.figure()
         for forecast in fnames:
             ax = plt.subplot(111)
             obs_df.loc[:,forecast].hist(ax=ax,color="0.5",alpha=0.5)
             ax.plot([obs.loc[forecast,"obsval"],obs.loc[forecast,"obsval"]],ax.get_ylim(),"r")
    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

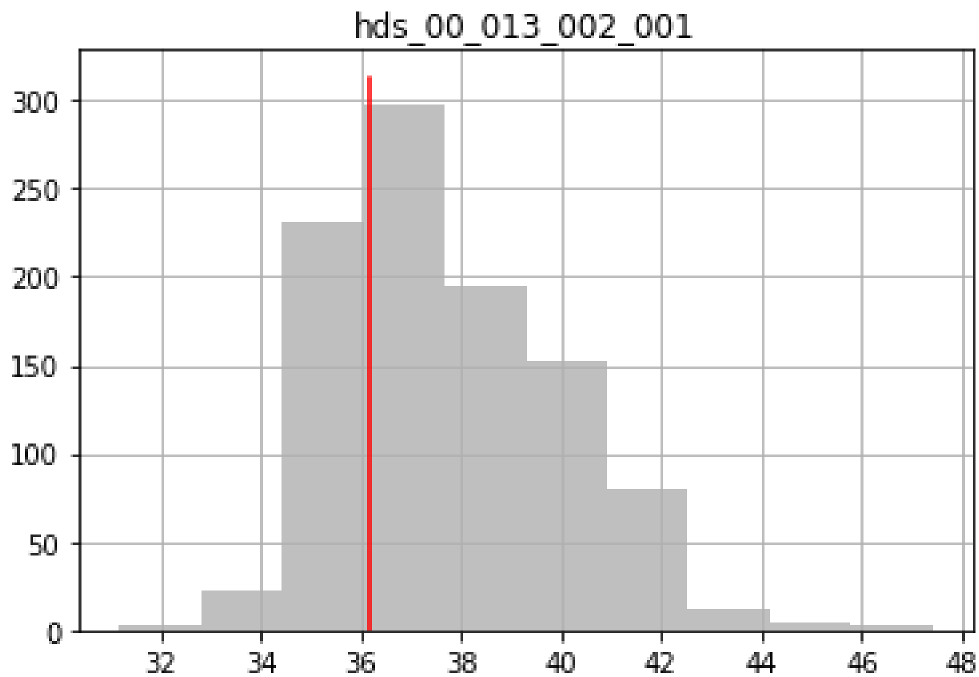

fa_hw_19791230
400
350
300
250
200
150
100
50
0
−1500
−1000
−500
0
500
1000

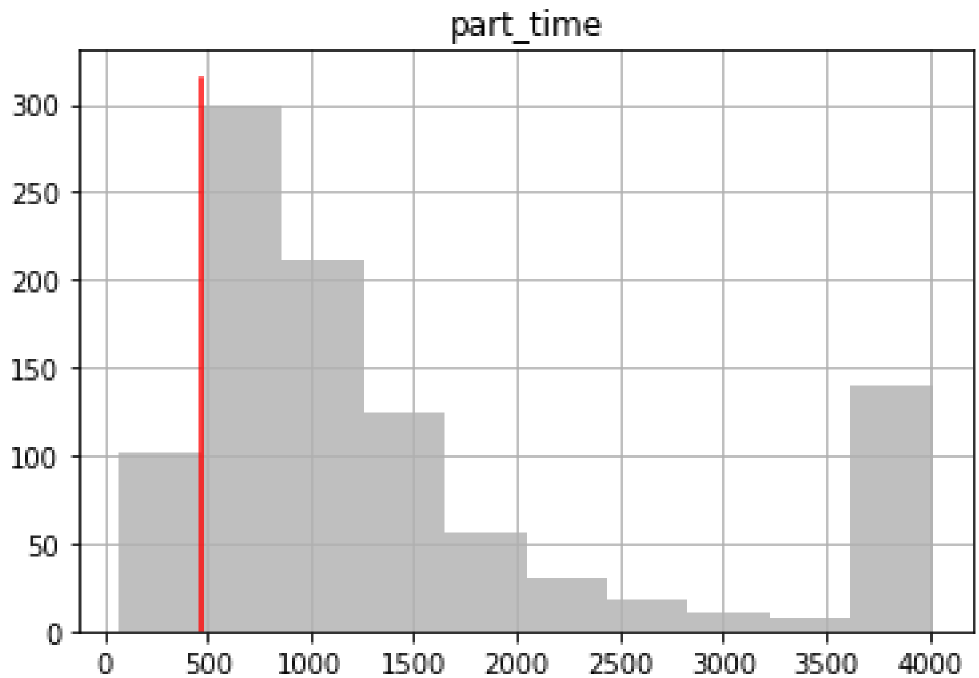

fa_hw_19801229
350
300
250
200
150
100
50
0
−1000
−500
0
500
1000
1500
2000

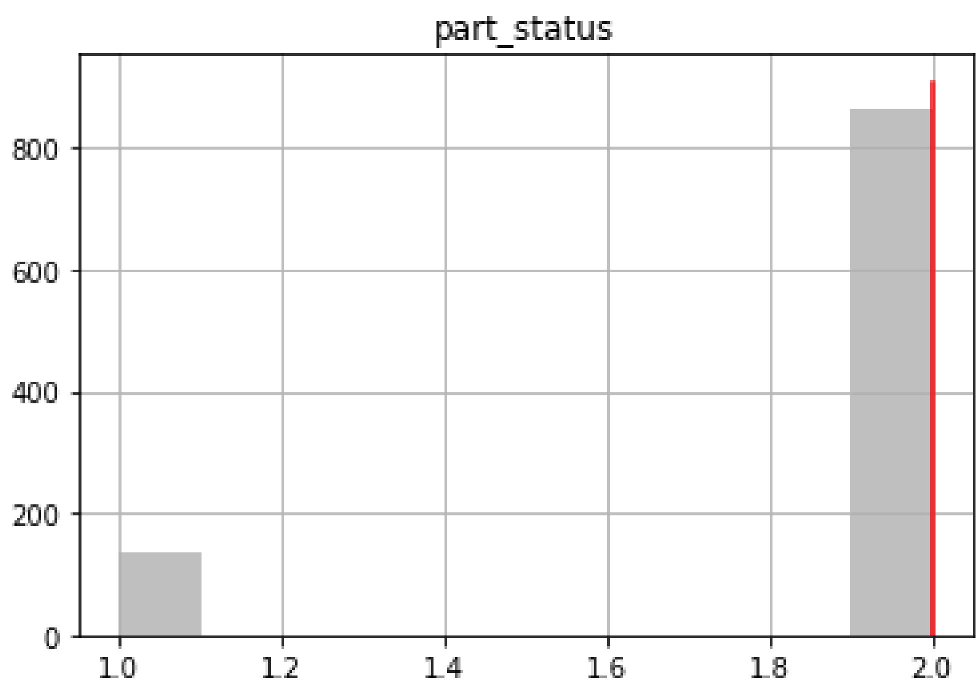

fa_tw_19791230
0
50
100
150
200
250
300
350
−1000
−500
0
500
1000
1500
2000
2500
3000


fa_tw_19801229
0
50
100
150
200
250
300
350
0
1000
2000
3000


hds_00_013_002_000
350
300
250
200
150
100
50
0
35.0
37.5
40.0
42.5
45.0
47.5
50.0
52.5
55.0

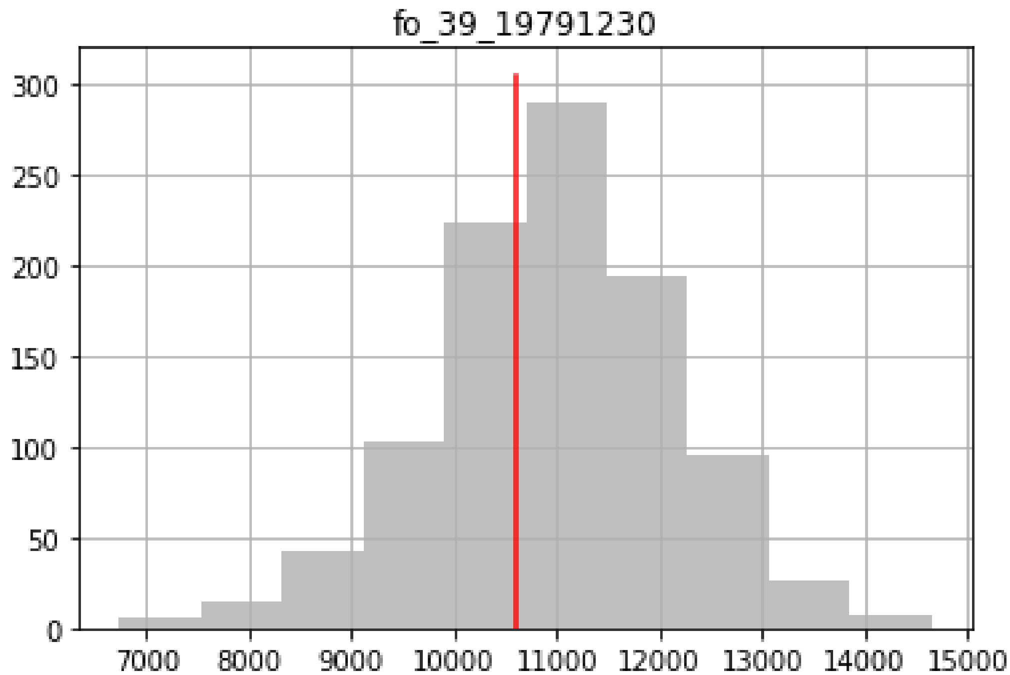

hds_00_013_002_001
300
250
200
150
100
50
0
32
34
36
38
40
42
44
46
48

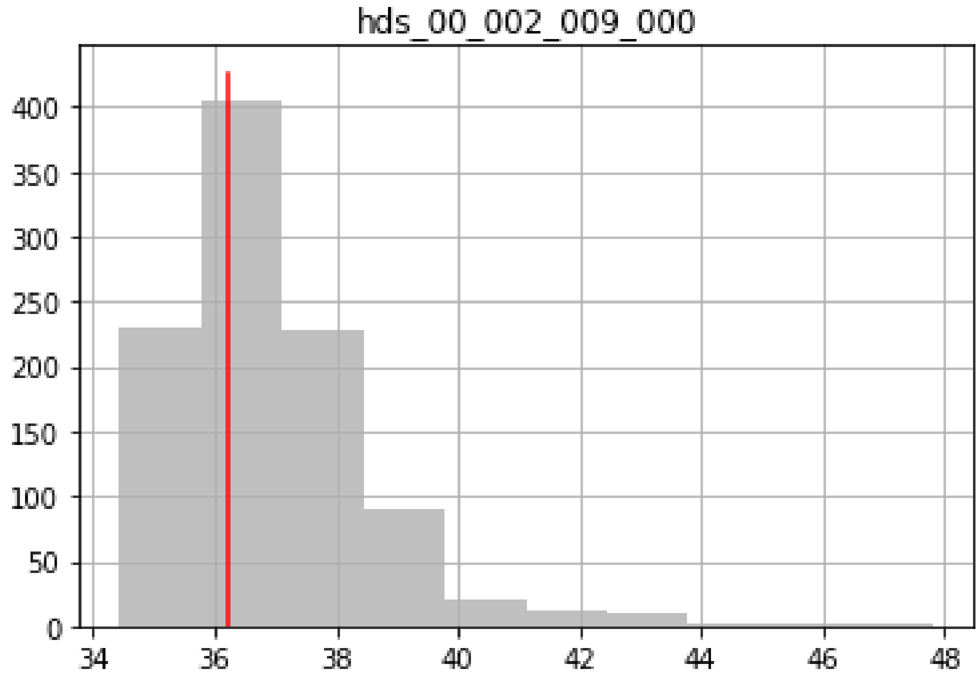


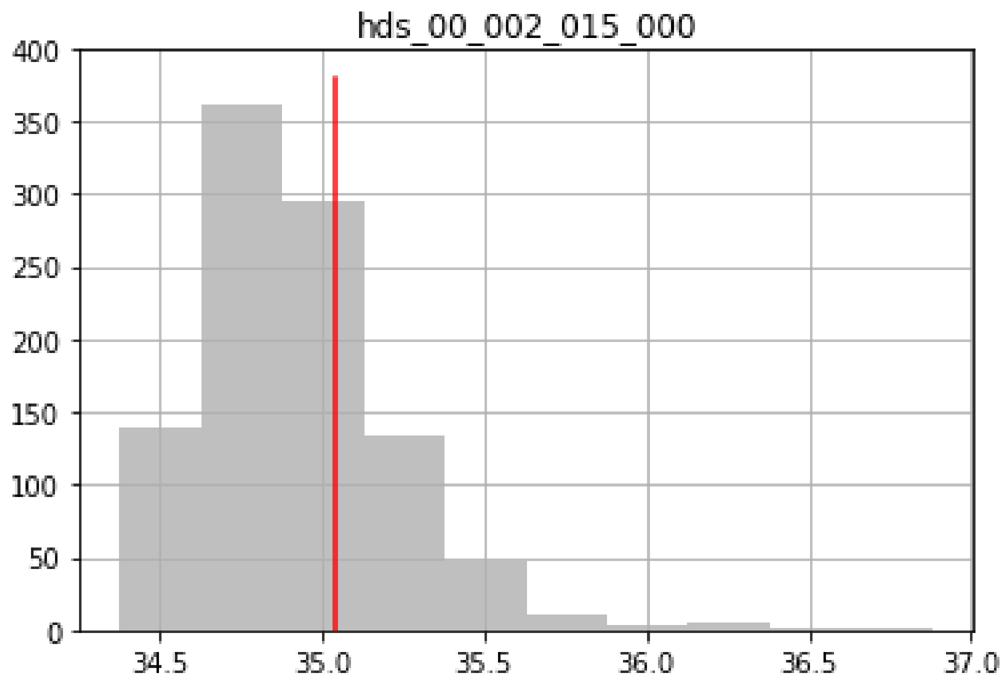


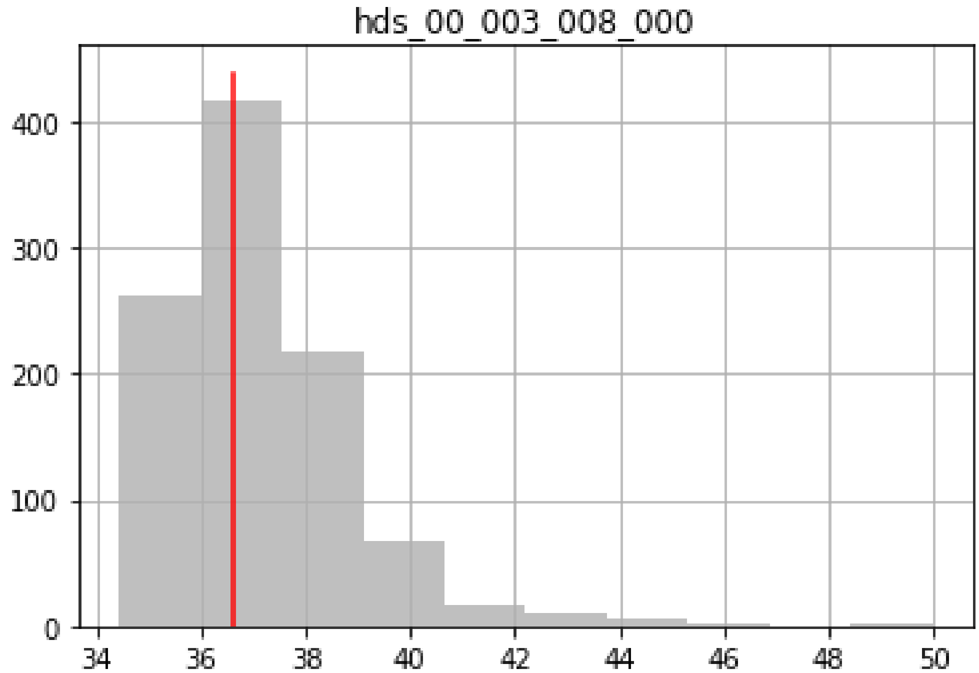
observations:

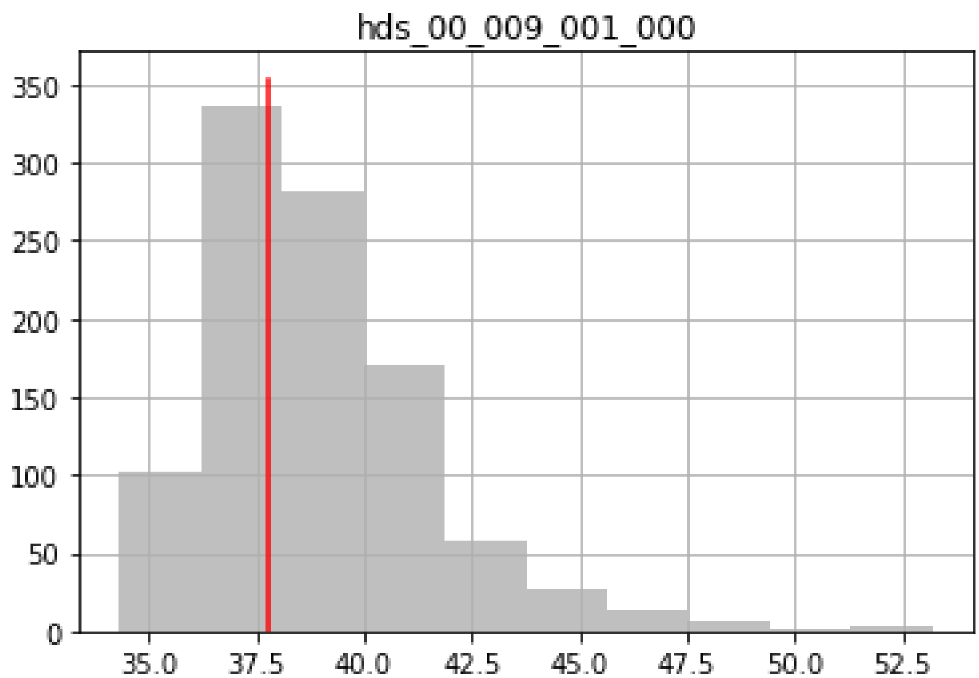
In [32]:
```
for oname in pst.nnz_obs_names:
    ax = plt.subplot(111)
    obs_df.loc[:,oname].hist(ax=ax,color="0.5",alpha=0.5)
    ax.plot([obs.loc[oname,"obsval"],obs.loc[oname,"obsval"]],ax.get_ylim(),"r")
    ax.set_title(oname)
    plt.show()
```

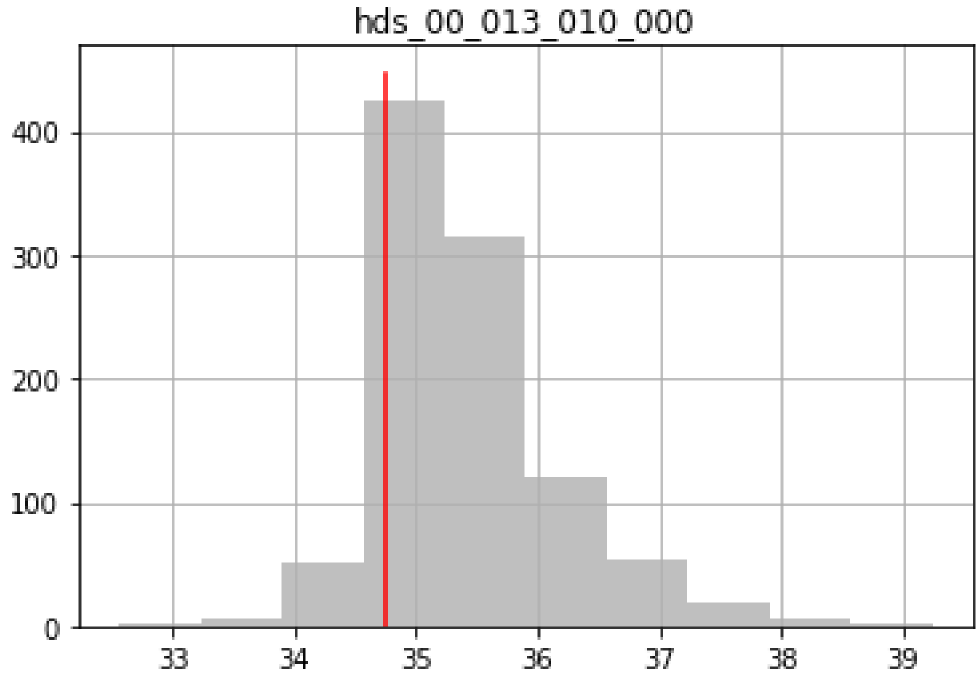


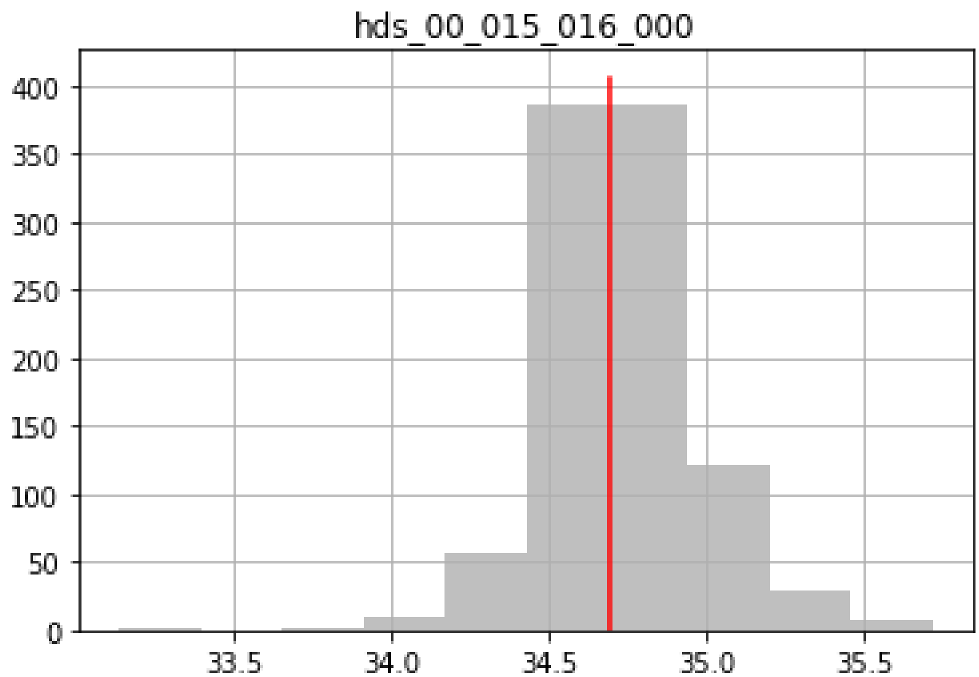

hds_00_002_009_000
0
50
100
150
200
250
300
350
400
34
36
38
40
42
44
46
48

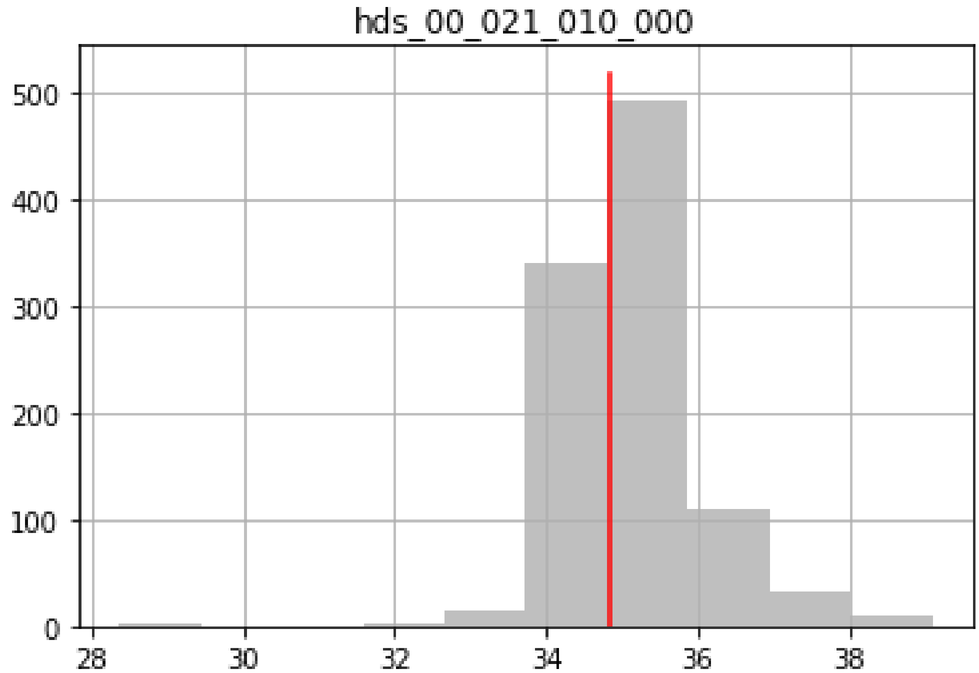

hds_00_002_015_000
0
50
100
150
200
250
300
350
400
34.5
35.0
35.5
36.0
36.5
37.0

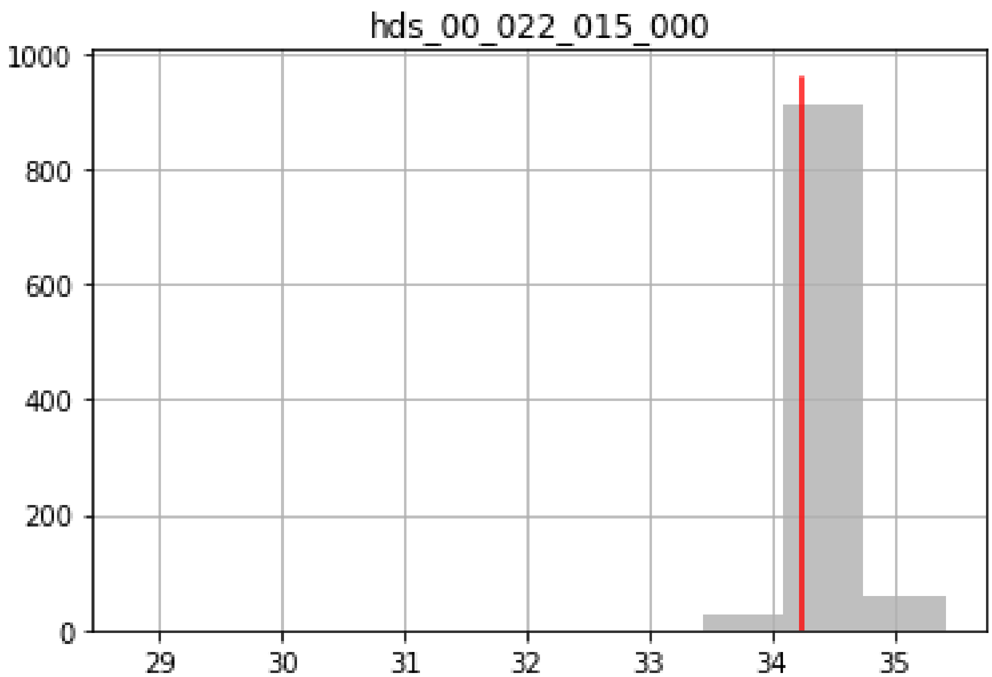

hds_00_003_008_000
400
300
200
100
0
34
36
38
40
42
44
46
48
50

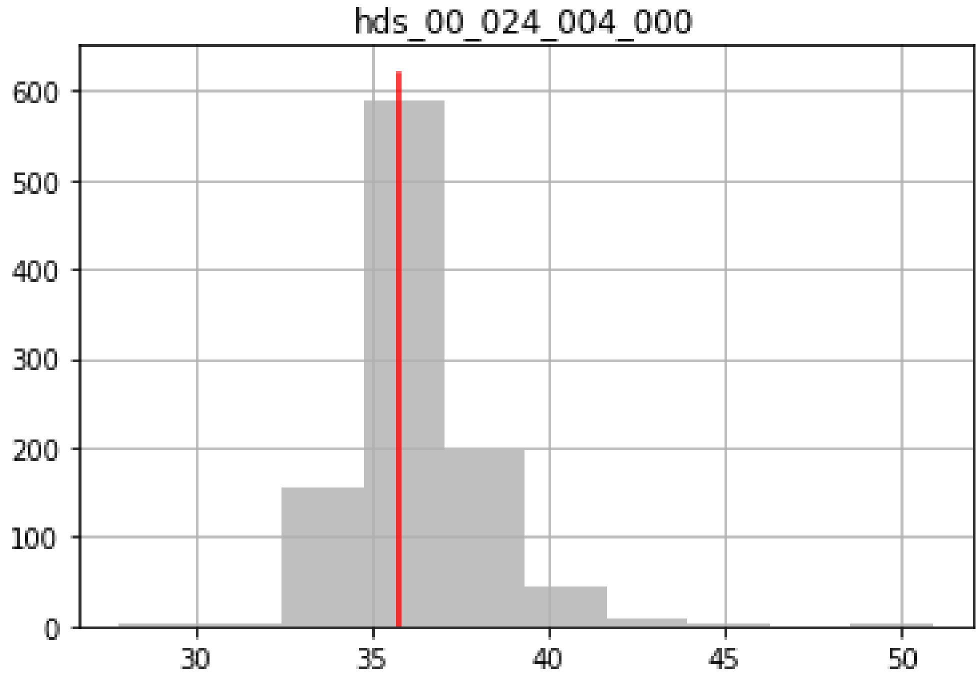

hds_00_009_001_000
350
300
250
200
150
100
50
0
35.0
37.5
40.0
42.5
45.0
47.5
50.0
52.5

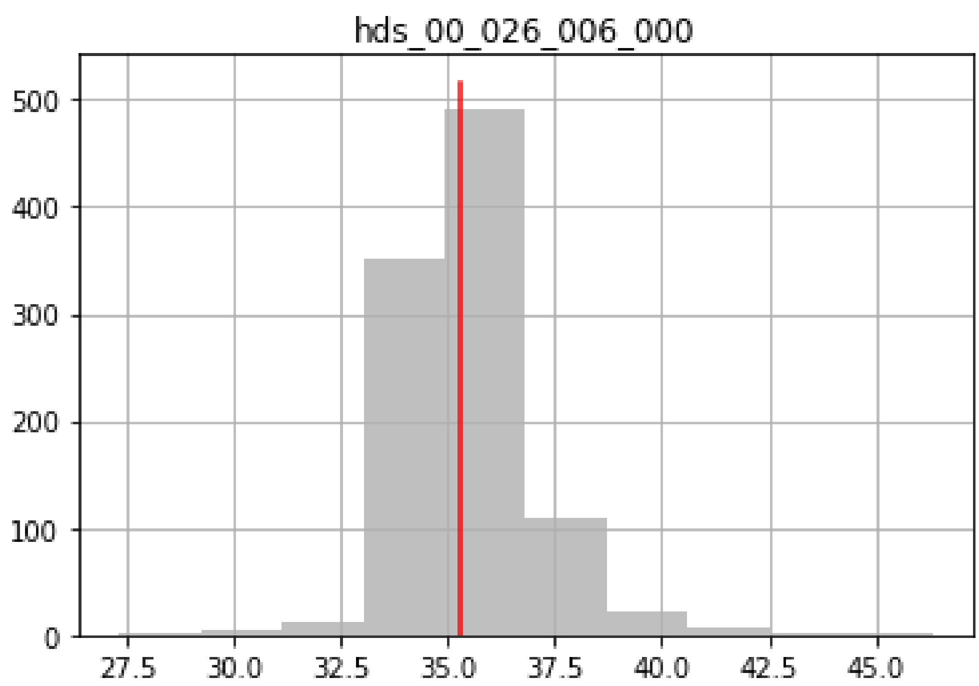

hds_00_013_010_000
400
300
200
100
0
33
34
35
36
37
38
39

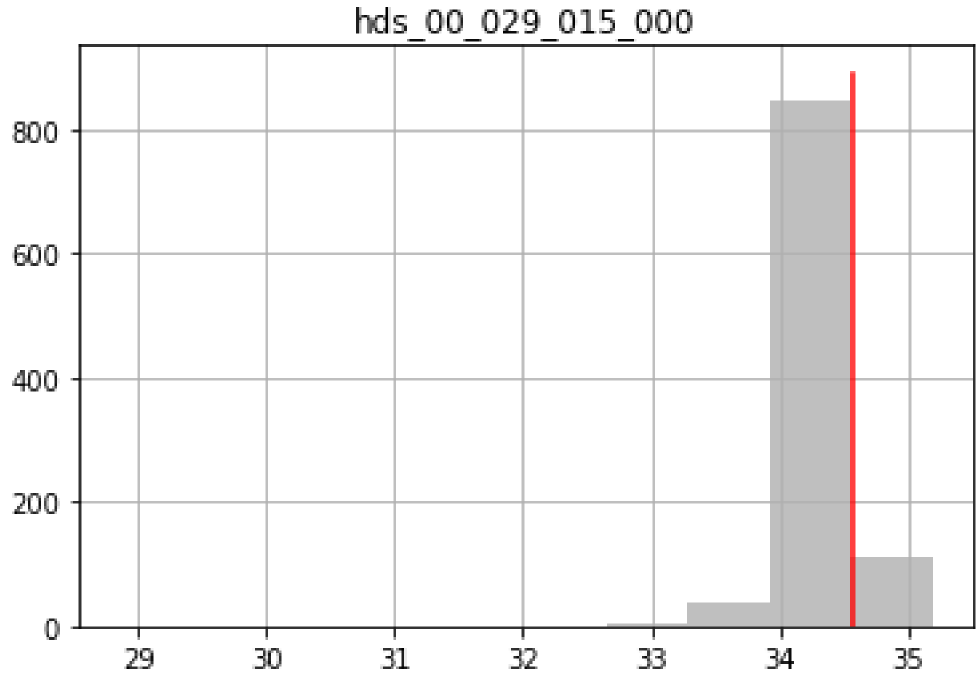

hds_00_015_016_000
400
350
300
250
200
150
100
50
0
33.5
34.0
34.5
35.0
35.5

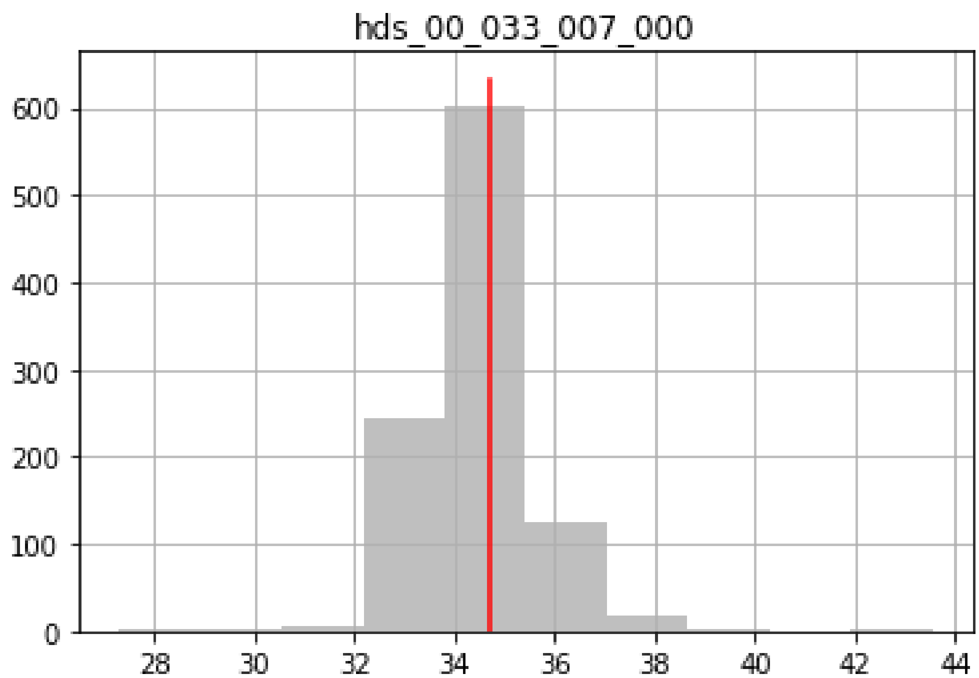

hds_00_021_010_000
500
400
300
200
100
0
28
30
32
34
36
38

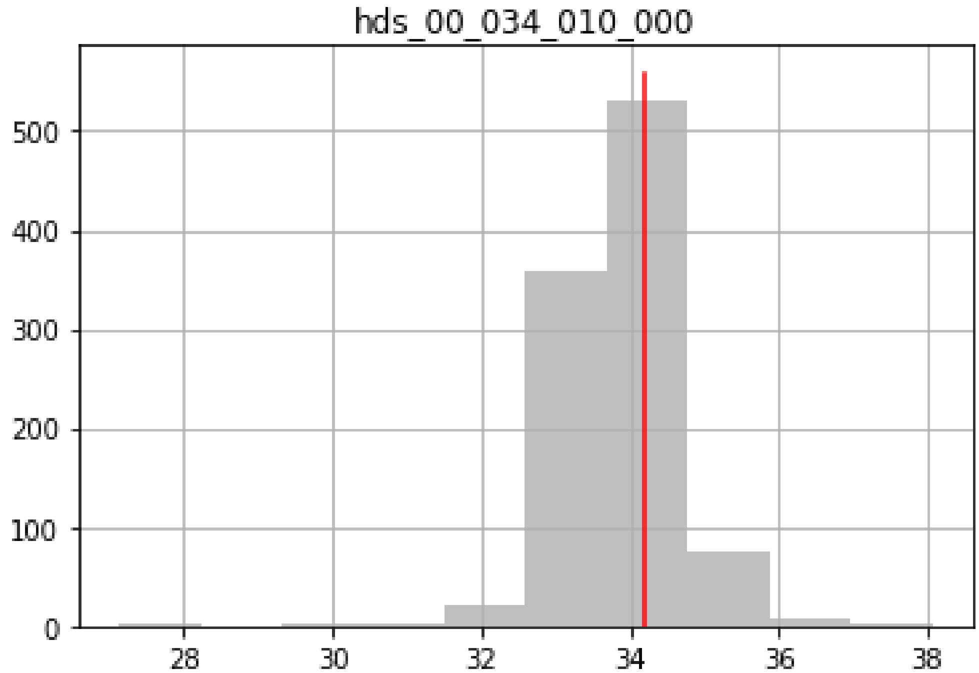

hds_00_022_015_000
1000
800
600
400
200
0
29
30
31
32
33
34
35


hds_00_024_004_000
600
500
400
300
200
100
0
30
35
40
45
50


hds_00_026_006_000
500
400
300
200
100
0
27.5
30.0
32.5
35.0
37.5
40.0
42.5
45.0


hds_00_029_015_000
800
600
400
200
0
29
30
31
32
33
34
35


hds_00_033_007_000
600
500
400
300
200
100
0
28
30
32
34
36
38
40
42
44


hds_00_034_010_000
500
400
300
200
100
0
28
30
32
34
36
38