pestpp-glm_part2

July 1, 2019

1 PESTPP-GLM Part 2

In this notebook, we will actually run PESTPP-GLM in standard parameter estimation mode and regularization mode. In both cases, we will use the baked-in bayes-linear posterior monte carlo analysis to get posterior forecast PDFs. We will use the prior monte carlo outputs as the prior forecast PDF. We will reuse the jacobian we used for FOSM to save runs

In [1]:
```
%matplotlib inline
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_csiro/notebooks/flopy
```

1.1 SUPER IMPORTANT: SET HOW MANY PARALLEL WORKERS TO USE

In [2]:
```
num_workers = 10
```

In [3]:
```
t_d = "template"
m_d = "master_glm_run"
```

In [4]:
```
pst = pyemu.Pst(os.path.join(t_d,"freyberg_pp.pst"))
pst.write_par_summary_table(filename="none")
```

Out[4]:
```
                     type transform  count  initial value upper bound  \
pp_prsity0    pp_prsity0       log     32              0     0.176091
strk                strk       log     40              0            2
pp_sy1            pp_sy1       log     32              0     0.243038
cn_prsity8    cn_prsity8       log      1              0     0.176091
gr_hk3             gr_hk3     fixed    705              1           10
cn_rech4         cn_rech4       log      1              0    0.0413927
```

pp_ss2	pp_ss2	log	32	0	1
gr_rech2	gr_rech2	fixed	705	1	1.1
gr_hk4	gr_hk4	fixed	705	1	10
pp_ss1	pp_ss1	log	32	0	1
cn_vka7	cn_vka7	log	1	0	1
pp_vka2	pp_vka2	fixed	32	1	10
cn_ss7	cn_ss7	log	1	0	1
cn_sy8	cn_sy8	log	1	0	0.243038
pp_rech1	pp_rech1	fixed	32	1	1.1
pp_prsity1	pp_prsity1	log	32	0	0.176091
gr_hk5	gr_hk5	fixed	705	1	10
gr_sy5	gr_sy5	fixed	705	1	1.75
gr_strt4	gr_strt4	fixed	705	1	1.05
cn_strt7	cn_strt7	log	1	0	0.0211893
pp_sy0	pp_sy0	log	32	0	0.243038
pp_vka0	pp_vka0	fixed	32	1	10
gr_prsity5	gr_prsity5	fixed	705	1	1.5
cn_rech5	cn_rech5	log	1	0	0.0413927
cn_strt6	cn_strt6	log	1	0	0.0211893
cn_ss6	cn_ss6	log	1	0	1
pp_strt2	pp_strt2	fixed	32	1	1.05
gr_ss3	gr_ss3	fixed	705	1	10
welflux	welflux	log	2	0	1
gr_strt5	gr_strt5	fixed	705	1	1.05
...	...	...	...	...	...
flow	flow	log	1	0	0.09691
pp_rech0	pp_rech0	log	32	0	0.0413927
pp_sy2	pp_sy2	log	32	0	0.243038
cn_hk8	cn_hk8	log	1	0	1
cn_hk7	cn_hk7	log	1	0	1
pp_ss0	pp_ss0	log	32	0	1
cn_strt8	cn_strt8	log	1	0	0.0211893
gr_sy3	gr_sy3	fixed	705	1	1.75
cn_sy7	cn_sy7	log	1	0	0.243038
pp_hk1	pp_hk1	log	32	0	1
pp_strt0	pp_strt0	fixed	32	1	1.05
gr_sy4	gr_sy4	fixed	705	1	1.75
pp_vka1	pp_vka1	log	32	0	1
pp_hk2	pp_hk2	log	32	0	1
gr_vka4	gr_vka4	fixed	705	1	10
pp_strt1	pp_strt1	fixed	32	1	1.05
gr_rech3	gr_rech3	fixed	705	1	1.1
gr_prsity4	gr_prsity4	fixed	705	1	1.5
gr_prsity3	gr_prsity3	fixed	705	1	1.5
welflux_k02	welflux_k02	log	6	0	1
cn_prsity7	cn_prsity7	log	1	0	0.176091
pp_prsity2	pp_prsity2	log	32	0	0.176091
cn_vka8	cn_vka8	log	1	0	1

cn_sy6	cn_sy6	log	1	0	0.243038
gr_vka3	gr_vka3	fixed	705	1	10
cn_hk6	cn_hk6	log	1	0	1
gr_vka5	gr_vka5	fixed	705	1	10
gr_ss4	gr_ss4	fixed	705	1	10
pp_hk0	pp_hk0	log	32	0	1
cn_vka6	cn_vka6	log	1	0	1

	lower bound	standard deviation
pp_prsity0	-0.30103	0.11928
strk	-2	1
pp_sy1	-0.60206	0.211275
cn_prsity8	-0.30103	0.11928
gr_hk3	0.1	2.475
cn_rech4	-0.0457575	0.0217875
pp_ss2	-1	0.5
gr_rech2	0.9	0.05
gr_hk4	0.1	2.475
pp_ss1	-1	0.5
cn_vka7	-1	0.5
pp_vka2	0.1	2.475
cn_ss7	-1	0.5
cn_sy8	-0.60206	0.211275
pp_rech1	0.9	0.05
pp_prsity1	-0.30103	0.11928
gr_hk5	0.1	2.475
gr_sy5	0.25	0.375
gr_strt4	0.95	0.025
cn_strt7	-0.0222764	0.0108664
pp_sy0	-0.60206	0.211275
pp_vka0	0.1	2.475
gr_prsity5	0.5	0.25
cn_rech5	-0.0457575	0.0217875
cn_strt6	-0.0222764	0.0108664
cn_ss6	-1	0.5
pp_strt2	0.95	0.025
gr_ss3	0.1	2.475
welflux	-1	0.5
gr_strt5	0.95	0.025
...	...	...
flow	-0.124939	0.0554622
pp_rech0	-0.0457575	0.0217875
pp_sy2	-0.60206	0.211275
cn_hk8	-1	0.5
cn_hk7	-1	0.5
pp_ss0	-1	0.5
cn_strt8	-0.0222764	0.0108664
gr_sy3	0.25	0.375

```
cn_sy7          -0.60206           0.211275
pp_hk1                -1                0.5
pp_strt0            0.95              0.025
gr_sy4              0.25              0.375
pp_vka1               -1                0.5
pp_hk2                -1                0.5
gr_vka4              0.1              2.475
pp_strt1            0.95              0.025
gr_rech3             0.9               0.05
gr_prsity4           0.5               0.25
gr_prsity3           0.5               0.25
welflux_k02           -1                0.5
cn_prsity7      -0.30103            0.11928
pp_prsity2      -0.30103            0.11928
cn_vka8               -1                0.5
cn_sy6          -0.60206           0.211275
gr_vka3              0.1              2.475
cn_hk6                -1                0.5
gr_vka5              0.1              2.475
gr_ss4               0.1              2.475
pp_hk0                -1                0.5
cn_vka6               -1                0.5

[65 rows x 7 columns]
```

Load and extract the portion of the prior we need for FOSM

```
In [5]: cov = pyemu.Cov.from_binary(os.path.join(t_d,"prior_cov.jcb"))
        cov.get(pst.adj_par_names).to_ascii(os.path.join(t_d,"glm_prior.cov"))
```

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

set some control options

```
In [6]: pst.control_data.noptmax = 3
        pst.pestpp_options["n_iter_base"] = -1
        pst.pestpp_options["n_iter_super"] = 3
        pst.pestpp_options["num_reals"] = 50 # this is how many ies uses
        pst.pestpp_options["parcov"] = "glm_prior.cov"
        pst.pestpp_options["base_jacobian"] = "freyberg_reuse.jcb"
        pst.write(os.path.join(t_d,"freyberg_pp.pst"))
```

```
noptmax:3, npar_adj:527, nnz_obs:14
```

```
In [7]: shutil.copy2(os.path.join("master_glm","freyberg_pp.jcb"),os.path.join(t_d,"freyberg_re
```

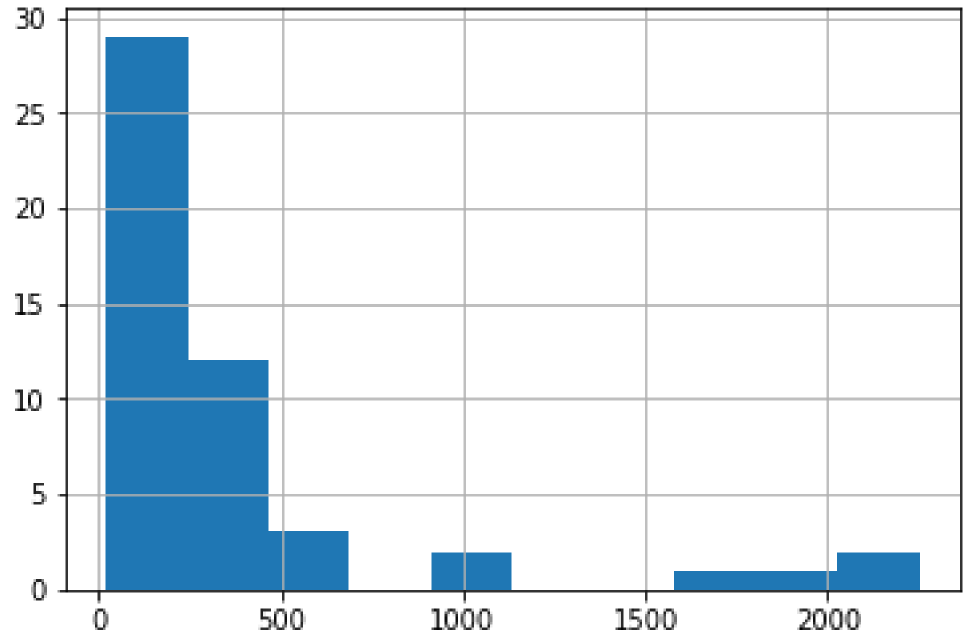
```
Out[7]: 'template/freyberg_reuse.jcb'
```

In [8]:
```
pyemu.os_utils.start_slaves(t_d,"pestpp-glm","freyberg_pp.pst",num_slaves=num_workers,s
                            master_dir=m_d)
```

In [9]:
```
df = df=pd.read_csv(os.path.join(m_d,"freyberg_pp.post.obsen.csv"),index_col=0)
oe = pyemu.ObservationEnsemble.from_dataframe(pst=pst,df=df)
```

In [10]:
```
ax = oe.phi_vector.hist()#bins=np.linspace(0,100,20))
oe.phi_vector.sort_values().iloc[:20]
```

Out[10]:
```
real_name
38     20.252546
15     21.726850
22     28.468621
5      40.663327
0      44.486239
1      46.335243
31     48.602332
41     50.931394
26     52.593058
6      53.499512
12     76.266670
18     81.114914
48     83.017110
2      88.304519
43     88.344489
10     89.887594
23     94.466519
24    105.041303
37    112.244877
49    116.235441
dtype: float64
```



Here we see the distribution of phi values across the 100 posterior realizations. Should we accept all of these??? The theoretical phi we should accept is number of nonzero obs (14).

To get a "posterior" ensemble, we need to throw out the realizations with large phi - lets just take the 20 best:

```
In [11]: oe_pt = oe.loc[oe.phi_vector.sort_values().index[:20],:] #just take the 20 lowest phi
```

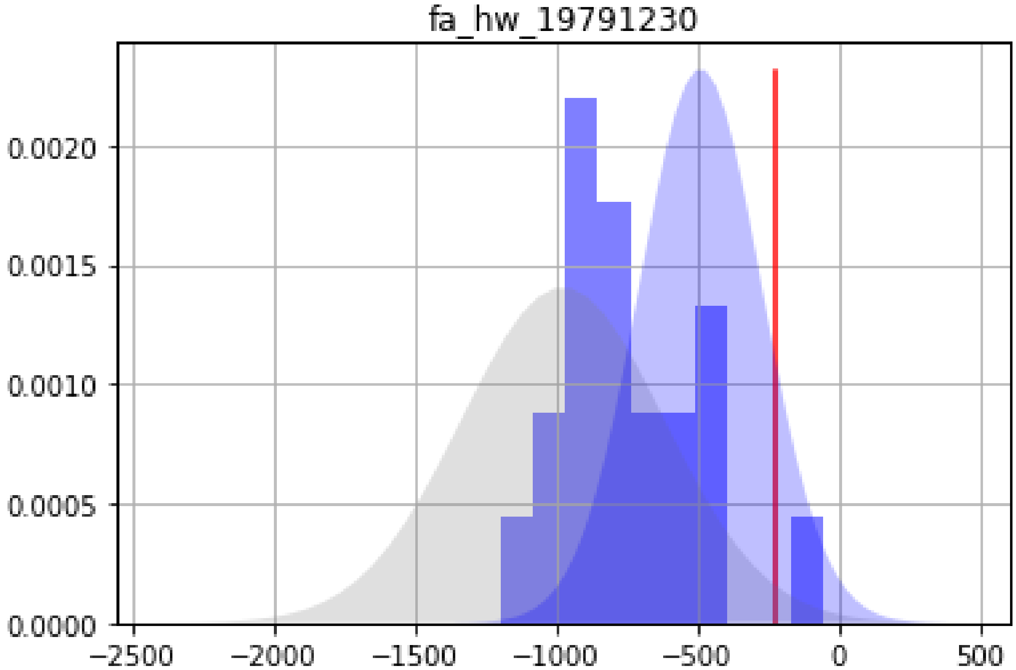
We can also load and plot the FOSM forecast results along side of the ensemble results:

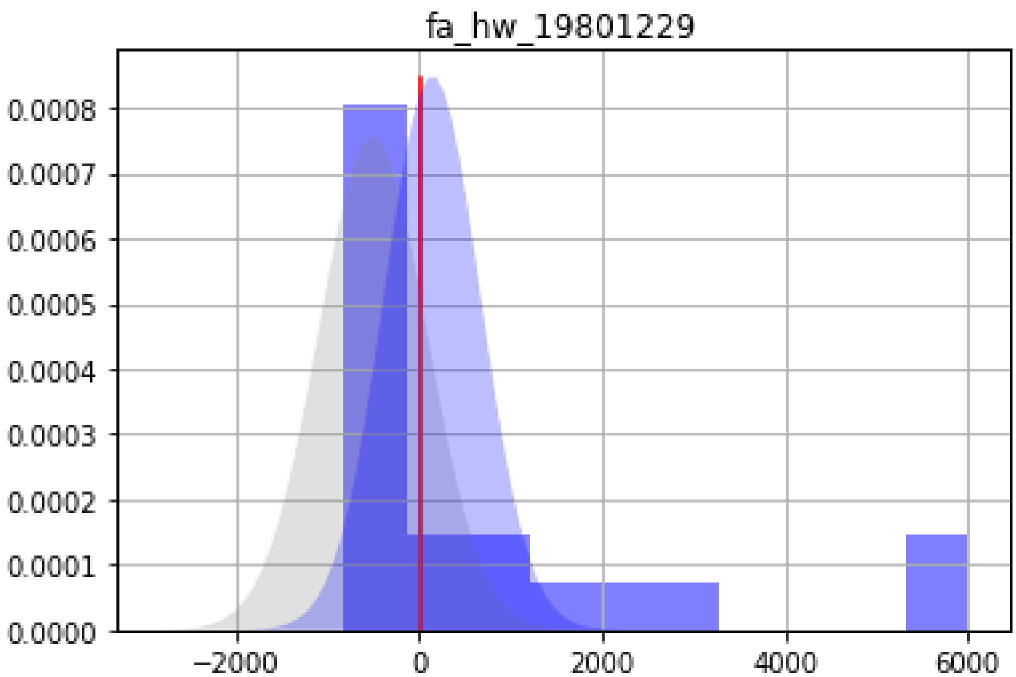
```
In [12]: f_df = pd.read_csv(os.path.join(m_d,"freyberg_pp.pred.usum.csv"),index_col=0)
         f_df.index = f_df.index.map(str.lower)
         f_df
```

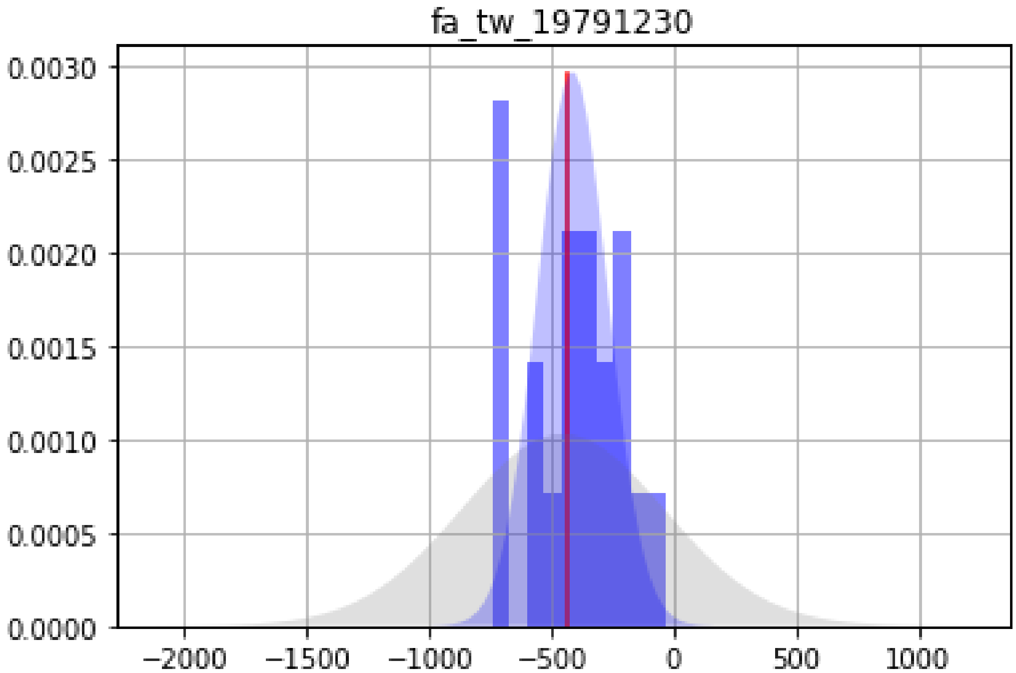
```
Out[12]:                     prior_mean  prior_stdev  prior_lower_bound  \
         name
         fa_hw_19791230       -977.2390    359.20800        -1695.65000
         fa_hw_19801229       -500.5320    589.30900        -1679.15000
         fa_tw_19791230       -453.0330    414.90700        -1282.85000
         fa_tw_19801229        -11.2061    610.23700        -1231.68000
         hds_00_013_002_000     39.6102      2.76268           34.08490
         hds_00_013_002_001     39.0079      2.91274           33.18250
         part_status             2.0000      0.00000            2.00000
         part_time             907.7020    458.56200           -9.42189

                             prior_upper_bound  post_mean  post_stdev  \
         name
         fa_hw_19791230              -258.8230  -489.5030  217.519000
```

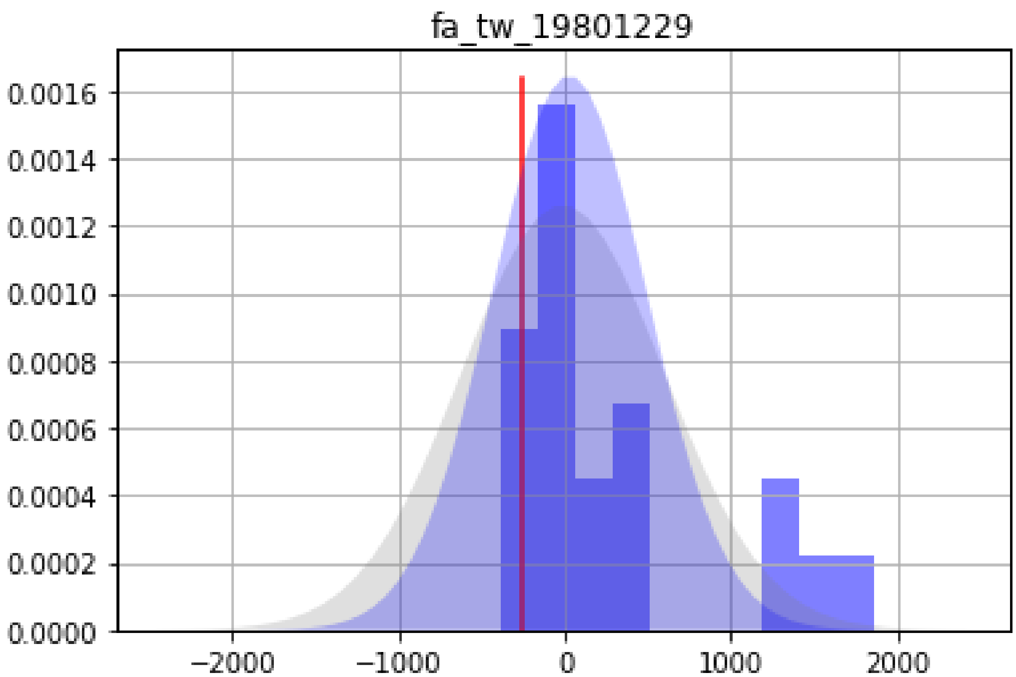
```
fa_hw_19801229              678.0860   149.9420  525.460000
fa_tw_19791230              376.7810  -413.4270  143.861000
fa_tw_19801229             1209.2700    28.3350  467.609000
hds_00_013_002_000           45.1356    37.7875    0.299594
hds_00_013_002_001           44.8334    37.1447    0.410993
part_status                   2.0000     2.0000    0.000000
part_time                  1824.8300   890.3290  336.786000

                    post_lower_bound  post_upper_bound
name
fa_hw_19791230              -924.5410         -54.4663
fa_hw_19801229              -900.9780        1200.8600
fa_tw_19791230              -701.1500        -125.7050
fa_tw_19801229              -906.8820         963.5520
hds_00_013_002_000            37.1883          38.3867
hds_00_013_002_001            36.3227          37.9667
part_status                    2.0000           2.0000
part_time                    216.7570        1563.9000
```

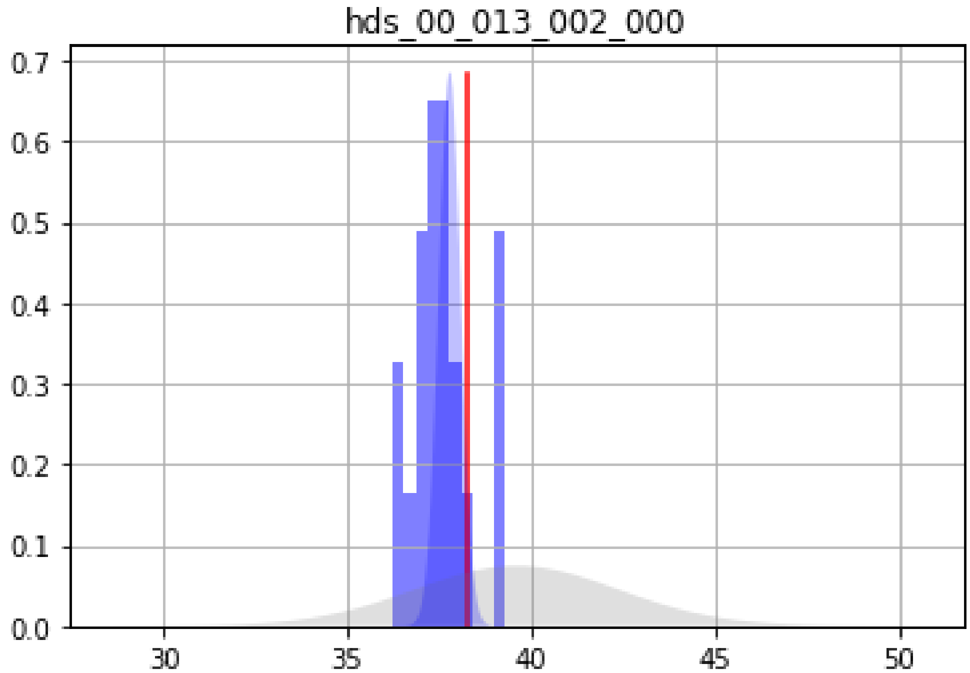
In [13]:
```
obs = pst.observation_data
fnames = pst.pestpp_options["forecasts"].split(",")
for forecast in fnames:
    ax = plt.subplot(111)
    oe_pt.loc[:,forecast].hist(ax=ax,color="b",alpha=0.5,normed=True)
    ax.plot([obs.loc[forecast,"obsval"],obs.loc[forecast,"obsval"]],ax.get_ylim(),"r"
    axt = plt.twinx()
    x,y = pyemu.plot_utils.gaussian_distribution(f_df.loc[forecast,"prior_mean"],f_df
    axt.fill_between(x,0,y,facecolor="0.5",alpha=0.25)
    x,y = pyemu.plot_utils.gaussian_distribution(f_df.loc[forecast,"post_mean"],f_df.
    axt.fill_between(x,0,y,facecolor="b",alpha=0.25)
    axt.set_ylim(0,axt.get_ylim()[1])
    axt.set_yticks([])
    ax.set_title(forecast)
    plt.show()
```

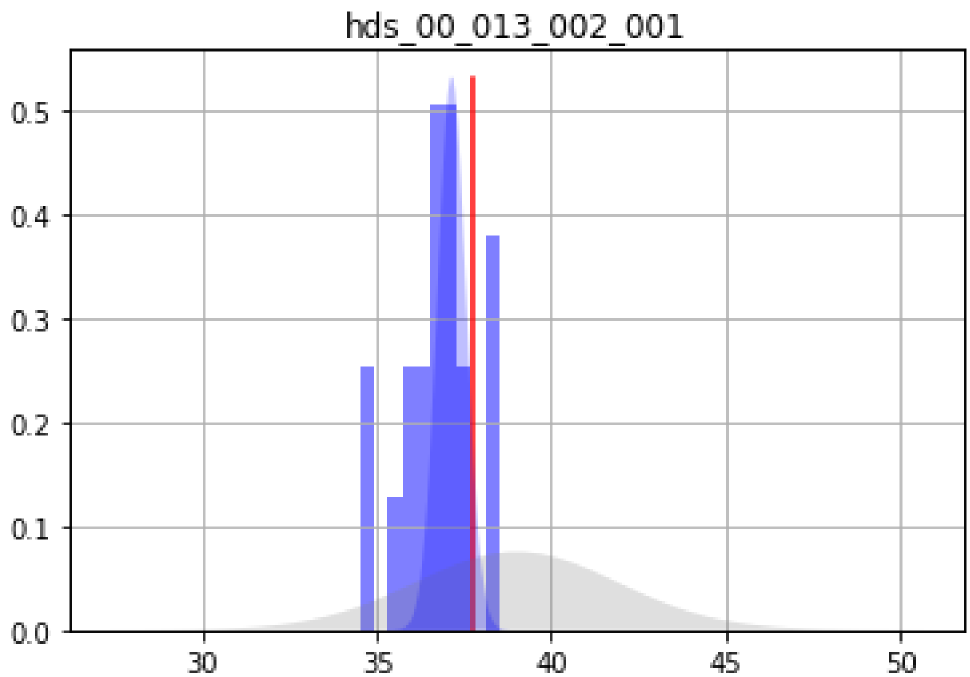

fa_hw_19791230
0.0000
0.0005
0.0010
0.0015
0.0020
−2500
−2000
−1500
−1000
−500
0
500

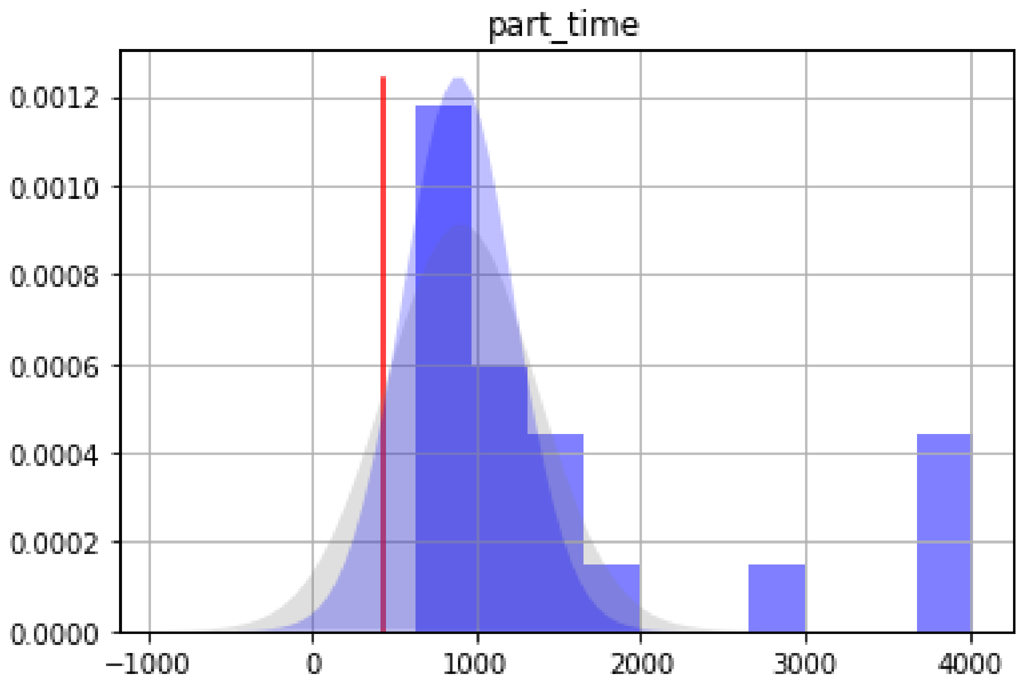

fa_hw_19801229
0.0000
0.0001
0.0002
0.0003
0.0004
0.0005
0.0006
0.0007
0.0008
−2000
0
2000
4000
6000

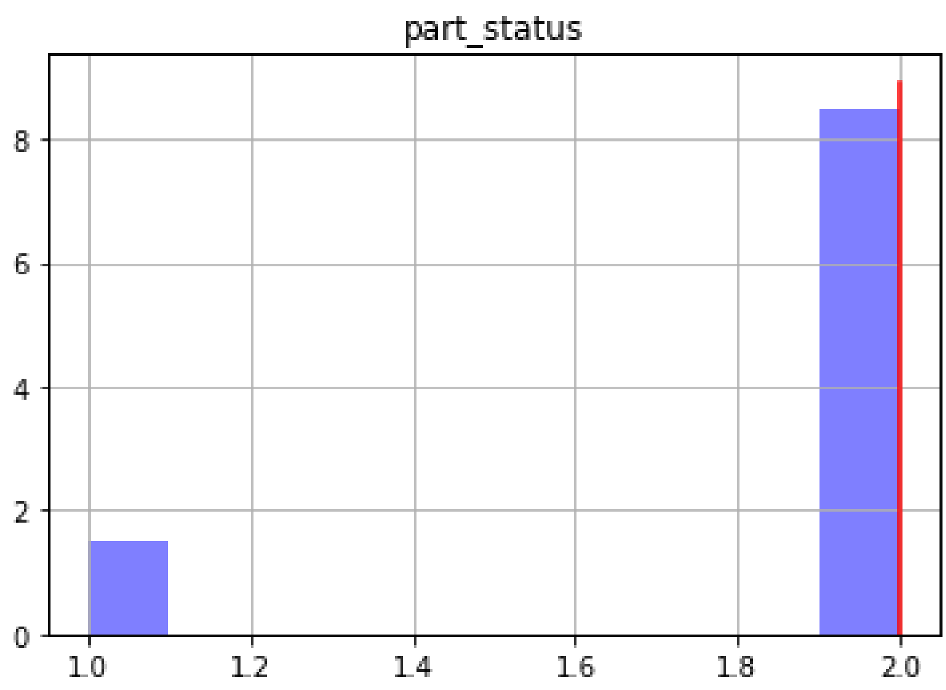

fa_tw_19791230
0.0030
0.0025
0.0020
0.0015
0.0010
0.0005
0.0000
−2000
−1500
−1000
−500
0
500
1000


fa_tw_19801229
0.0016
0.0014
0.0012
0.0010
0.0008
0.0006
0.0004
0.0002
0.0000
−2000
−1000
0
1000
2000


hds_00_013_002_000
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
30
35
40
45
50


hds_00_013_002_001
0.5
0.4
0.3
0.2
0.1
0.0
30
35
40
45
50


part_time
0.0012
0.0010
0.0008
0.0006
0.0004
0.0002
0.0000
−1000
0
1000
2000
3000
4000


part_status
8
6
4
2
0
1.0
1.2
1.4
1.6
1.8
2.0

1.1.1 Setup of Tikhonov regularization

Now lets setup and use some formal regularization to bring the final phi up to around 14. We will use first-order regularization based on the covariance matrix we build earlier:

```
In [14]: cov = pyemu.Cov.from_ascii(os.path.join(t_d,"glm_prior.cov"))
```

```
In [15]: pyemu.helpers.first_order_pearson_tikhonov(pst,cov)
```

```
getting CC matrix
processing
```

```
In [16]: pst.prior_information.sort_values(by="weight",ascending=False).head()
```

```
Out[16]:                                                           equation    obgnme  \
         pilbl
         pcc_1   1.0 * log(dc0000390005) - 1.0 * log(dc0000390006) = 0.0  regul_cc
         pcc_10  1.0 * log(dc0000390006) - 1.0 * log(dc0000390007) = 0.0  regul_cc
         pcc_43  1.0 * log(dc0000390012) - 1.0 * log(dc0000390013) = 0.0  regul_cc
         pcc_40  1.0 * log(dc0000390011) - 1.0 * log(dc0000390012) = 0.0  regul_cc
         pcc_36  1.0 * log(dc0000390010) - 1.0 * log(dc0000390011) = 0.0  regul_cc

                  pilbl    weight
         pilbl
         pcc_1    pcc_1  0.904837
         pcc_10  pcc_10  0.904837
         pcc_43  pcc_43  0.904837
         pcc_40  pcc_40  0.904837
         pcc_36  pcc_36  0.904837
```

```
In [17]: pst.reg_data.phimlim = pst.nnz_obs
         pst.reg_data.phimaccept = pst.reg_data.phimlim * 1.1
         pst.write(os.path.join(t_d,"freyberg_pp.pst"))
```

```
noptmax:3, npar_adj:527, nnz_obs:14
```

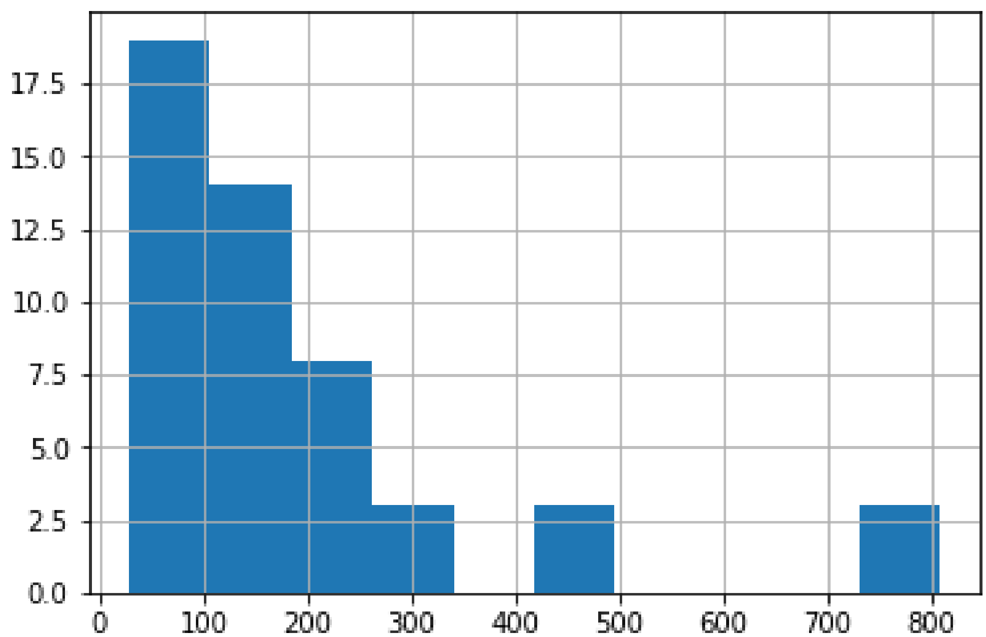
```
In [18]: pyemu.os_utils.start_slaves(t_d,"pestpp-glm","freyberg_pp.pst",num_slaves=num_workers
                                    master_dir=m_d)
```

```
In [19]: df = df=pd.read_csv(os.path.join(m_d,"freyberg_pp.post.obsen.csv"),index_col=0)
         oe = pyemu.ObservationEnsemble.from_dataframe(pst=pst,df=df)
```

```
In [20]: ax = oe.phi_vector.hist()#bins=np.linspace(0,100,20))
         oe.phi_vector.sort_values().iloc[:20]
```

```
Out[20]: real_name
         38     28.033506
         15     31.285081
```

```
43     37.305393
22     37.328963
0      38.684235
35     40.693600
31     43.121395
1      44.106285
5      49.328534
12     50.396984
48     55.666585
18     59.618114
46     62.342003
42     64.014081
44     66.322496
26     69.195631
6      79.017769
41     97.523164
37    100.319933
2     108.258736
dtype: float64
```



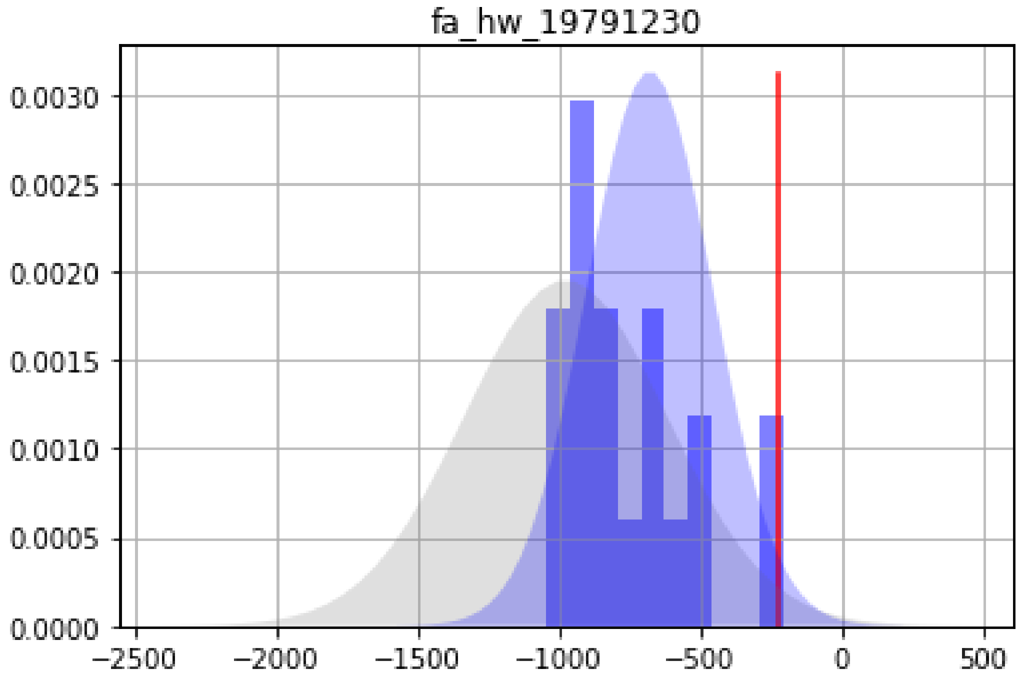
Same as before, to get a "posterior" ensemble, we need to throw out the realizations with large phi - lets just take the 20 best:

In [21]:
```
oe_pt = oe.loc[oe.phi_vector.sort_values().index[:20],:]
```

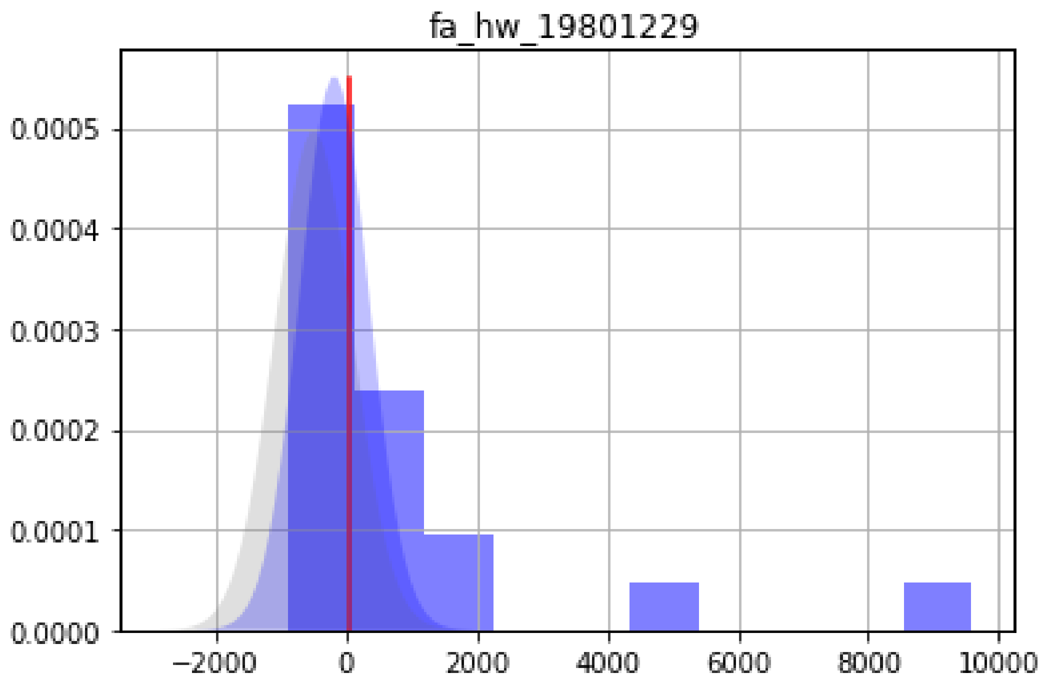
In [22]:
```
f_df = pd.read_csv(os.path.join(m_d,"freyberg_pp.pred.usum.csv"),index_col=0)
f_df.index = f_df.index.map(str.lower)
f_df
```

Out[22]:
```
                     prior_mean  prior_stdev  prior_lower_bound  \
name
fa_hw_19791230        -977.2390    359.20800        -1695.65000
fa_hw_19801229        -500.5320    589.30900        -1679.15000
fa_tw_19791230        -453.0330    414.90700        -1282.85000
fa_tw_19801229         -11.2061    610.23700        -1231.68000
hds_00_013_002_000      39.6102      2.76268           34.08490
hds_00_013_002_001      39.0079      2.91274           33.18250
part_status              2.0000      0.00000            2.00000
part_time              907.7020    458.56200           -9.42189

                     prior_upper_bound  post_mean  post_stdev  \
name
fa_hw_19791230               -258.8230  -679.2560  223.932000
fa_hw_19801229                678.0860  -184.0380  531.386000
fa_tw_19791230                376.7810  -387.6090  145.300000
fa_tw_19801229               1209.2700    -5.3833  468.053000
hds_00_013_002_000             45.1356    37.8504    0.299934
hds_00_013_002_001             44.8334    37.4613    0.411130
part_status                     2.0000     2.0000    0.000000
part_time                    1824.8300  1034.0000  337.515000

                     post_lower_bound  post_upper_bound
name
fa_hw_19791230             -1127.1200         -231.3920
fa_hw_19801229             -1246.8100          878.7330
fa_tw_19791230              -678.2080          -97.0099
fa_tw_19801229              -941.4900          930.7230
hds_00_013_002_000            37.2506           38.4503
hds_00_013_002_001            36.6390           38.2836
part_status                    2.0000            2.0000
part_time                    358.9680         1709.0300
```

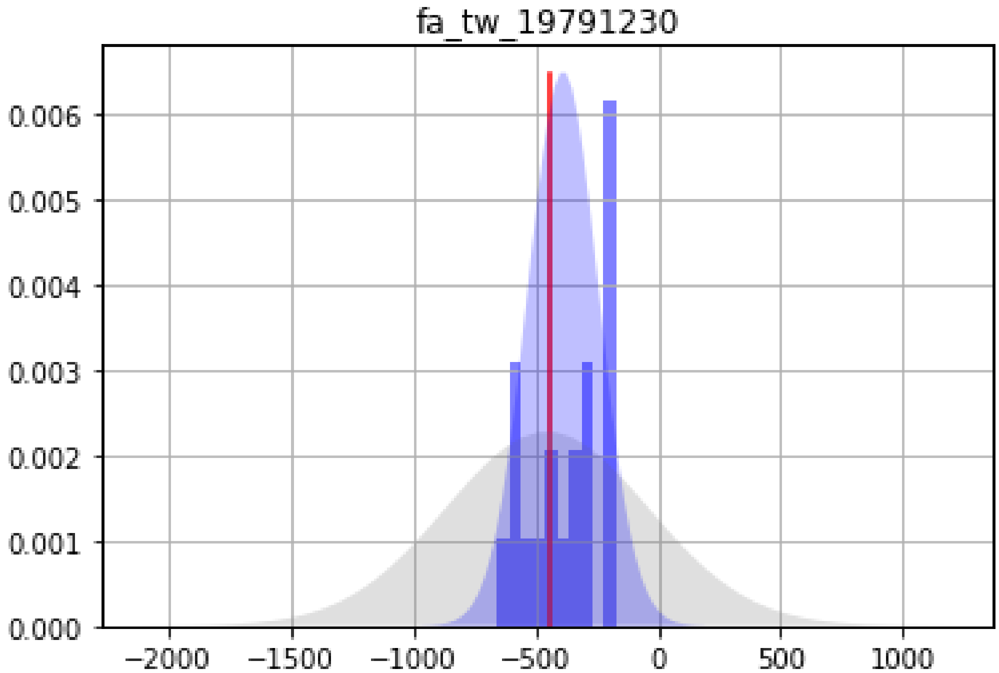
In [23]:
```
obs = pst.observation_data
fnames = pst.pestpp_options["forecasts"].split(",")
for forecast in fnames:
    ax = plt.subplot(111)
    oe_pt.loc[:,forecast].hist(ax=ax,color="b",alpha=0.5,normed=True)
    ax.plot([obs.loc[forecast,"obsval"],obs.loc[forecast,"obsval"]],ax.get_ylim(),"r"
    axt = plt.twinx()
    x,y = pyemu.plot_utils.gaussian_distribution(f_df.loc[forecast,"prior_mean"],f_df
    axt.fill_between(x,0,y,facecolor="0.5",alpha=0.25)
    x,y = pyemu.plot_utils.gaussian_distribution(f_df.loc[forecast,"post_mean"],f_df.
    axt.fill_between(x,0,y,facecolor="b",alpha=0.25)
    axt.set_ylim(0,axt.get_ylim()[1])
    axt.set_yticks([])
    ax.set_title(forecast)
    plt.show()
```

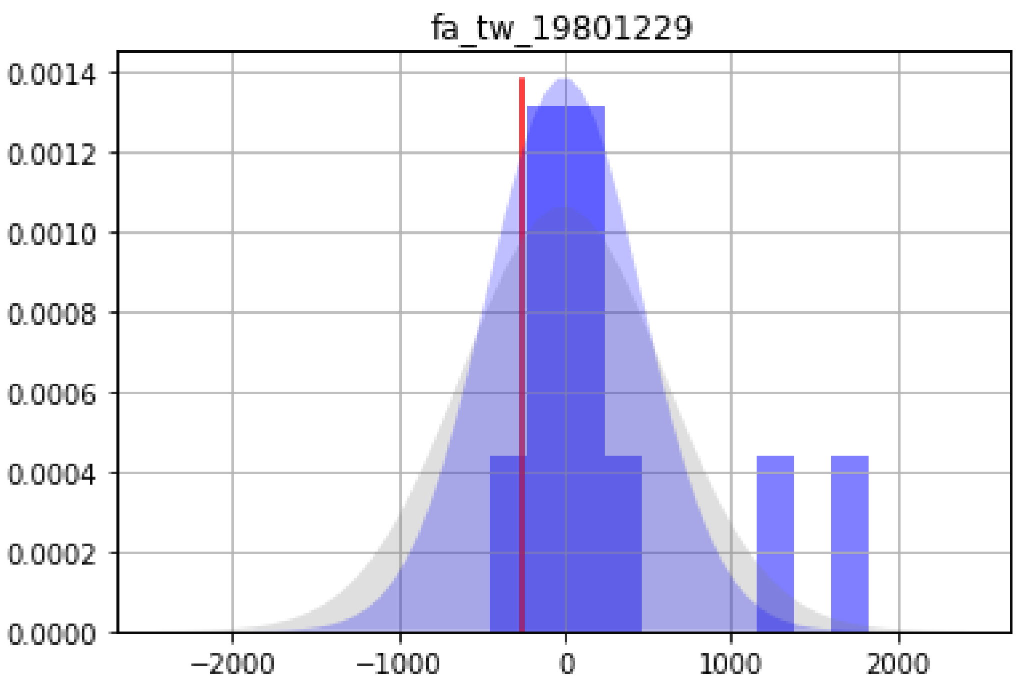


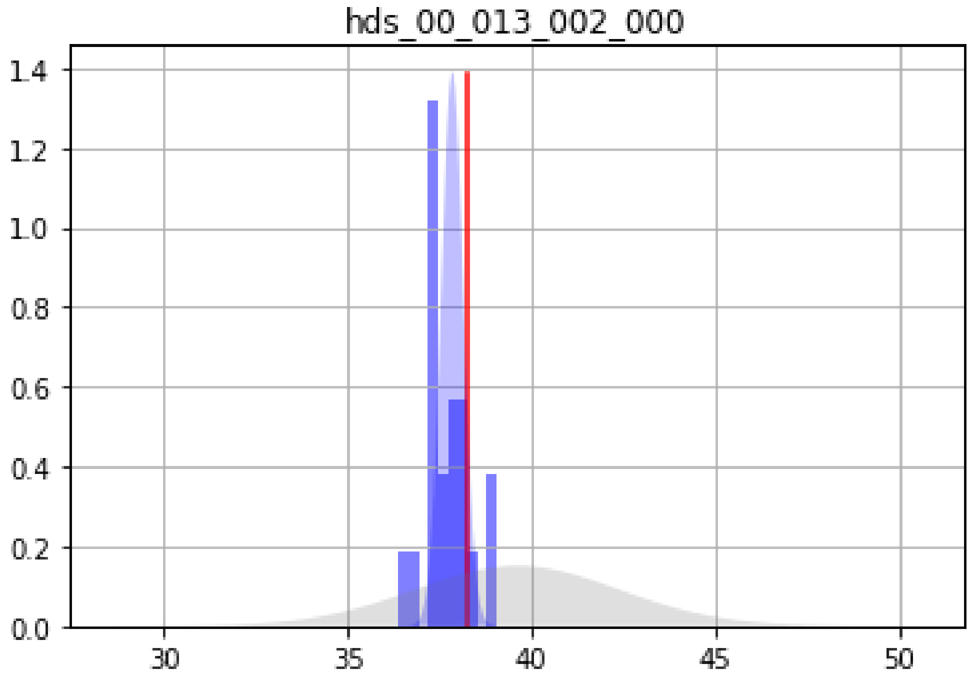
fa_hw_19791230
0.0030
0.0025
0.0020
0.0015
0.0010
0.0005
0.0000
−2500
−2000
−1500
−1000
−500
0
500

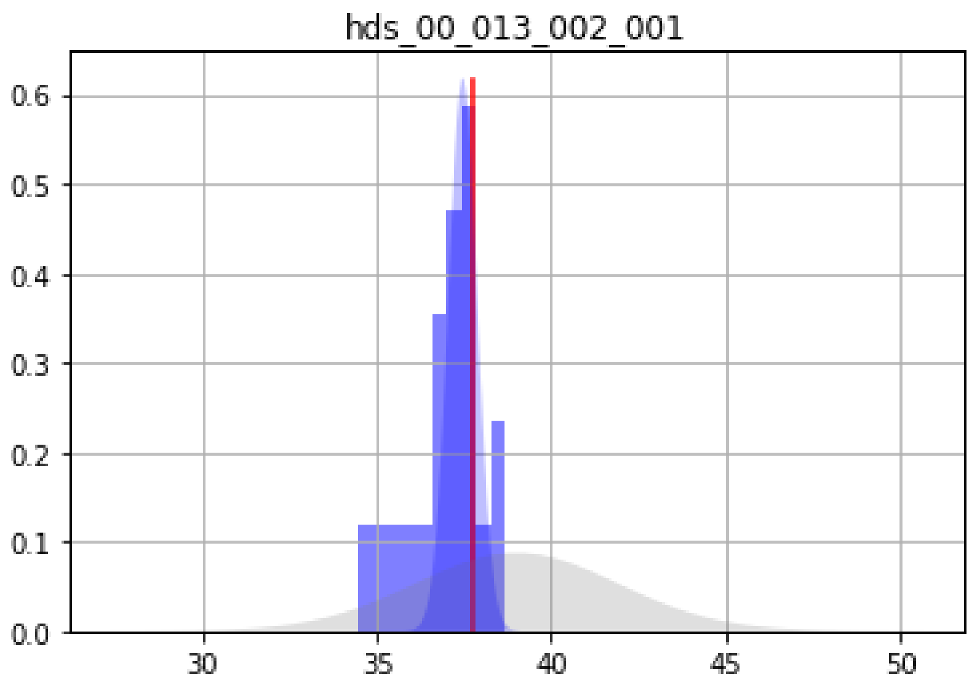


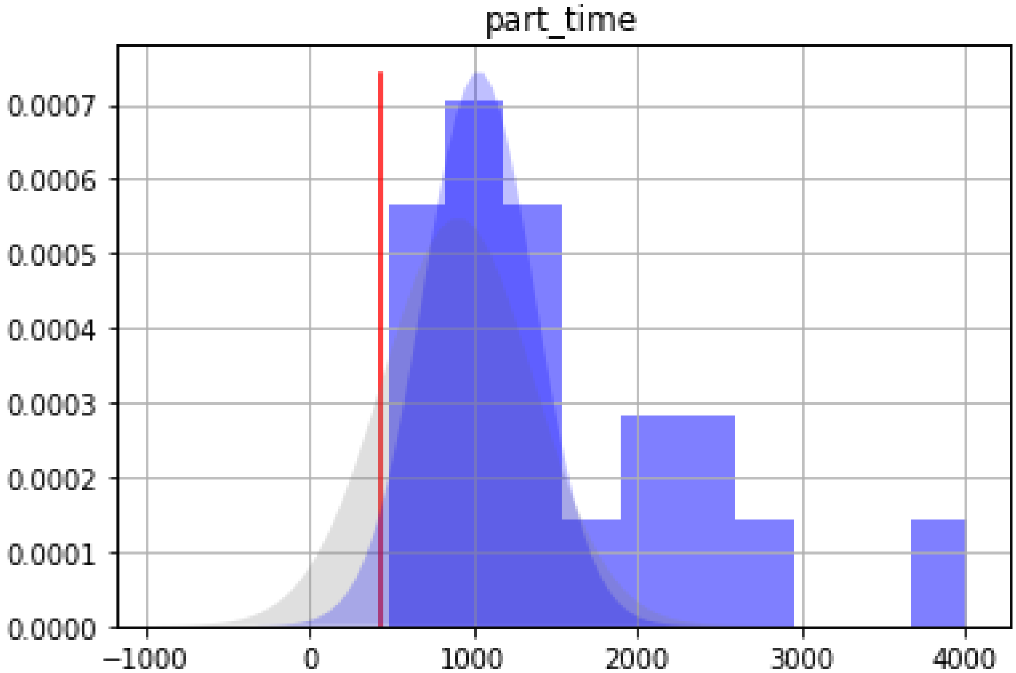
fa_hw_19801229
0.0005
0.0004
0.0003
0.0002
0.0001
0.0000
−2000
0
2000
4000
6000
8000
10000

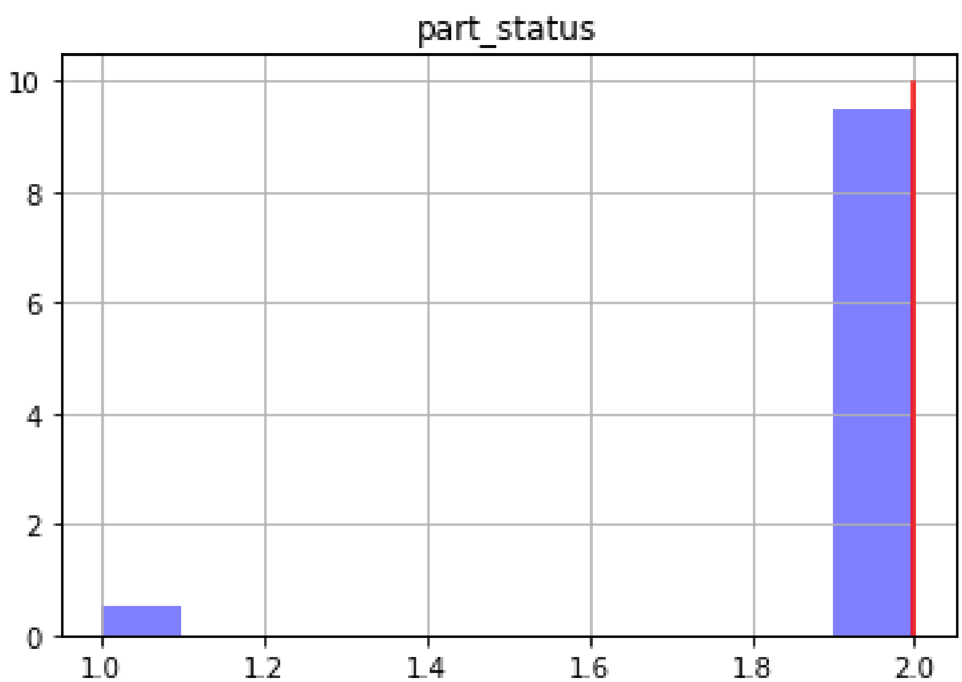

fa_tw_19791230
0.006
0.005
0.004
0.003
0.002
0.001
0.000
−2000
−1500
−1000
−500
0
500
1000


fa_tw_19801229
0.0014
0.0012
0.0010
0.0008
0.0006
0.0004
0.0002
0.0000
−2000
−1000
0
1000
2000


hds_00_013_002_000
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
30
35
40
45
50


hds_00_013_002_001
0.6
0.5
0.4
0.3
0.2
0.1
0.0
30
35
40
45
50


part_time
0.0007
0.0006
0.0005
0.0004
0.0003
0.0002
0.0001
0.0000
−1000
0
1000
2000
3000
4000


part_status
10
8
6
4
2
0
1.0
1.2
1.4
1.6
1.8
2.0