dataworth_worked

July 18, 2019

1 data worth and related assessments

In this notebook, we will use outputs from previous notebooks (in particular `pestpp-glm_part1.ipynb`) to undertake data worth assessments based on first-order second-moment (FOSM) techniques. "Worth" is framed here in the context of the extent to which the uncertainty surrounding a model prediction of management interest is reduced through data collection. Given that these anayses can help target and optimize data acquisition strategies, this is a concept that really resonates with decision makers.

```
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 C:\Users\knowling\Dev\GW1876\activities_csiro\notebooks\flopy
```

```
In [2]: m_d = "master_glm"
```

```
In [3]: pst = pyemu.Pst(os.path.join(m_d,"freyberg_pp.pst"))
        print(pst.npar_adj)
        pst.write_par_summary_table(filename="none")
```

```
527
```

```
Out[3]:                          type transform  count initial value upper bound  \
        cn_hk6              cn_hk6       log      1             0           1
        cn_hk7              cn_hk7       log      1             0           1
        cn_hk8              cn_hk8       log      1             0           1
        cn_prsity6     cn_prsity6       log      1             0   0.0791812
        cn_prsity7     cn_prsity7       log      1             0   0.0791812
```

cn_prsity8	cn_prsity8	log	1	0	0.0791812
cn_rech4	cn_rech4	log	1	0	0.0413927
cn_rech5	cn_rech5	log	1	0	0.0413927
cn_ss6	cn_ss6	log	1	0	1
cn_ss7	cn_ss7	log	1	0	1
cn_ss8	cn_ss8	log	1	0	1
cn_strt6	cn_strt6	log	1	0	0.0211893
cn_strt7	cn_strt7	log	1	0	0.0211893
cn_strt8	cn_strt8	log	1	0	0.0211893
cn_sy6	cn_sy6	log	1	0	0.243038
cn_sy7	cn_sy7	log	1	0	0.243038
cn_sy8	cn_sy8	log	1	0	0.243038
cn_vka6	cn_vka6	log	1	0	1
cn_vka7	cn_vka7	log	1	0	1
cn_vka8	cn_vka8	log	1	0	1
drncond_k00	drncond_k00	log	10	0	1
flow	flow	log	1	0	0.09691
gr_hk3	gr_hk3	fixed	705	1	10
gr_hk4	gr_hk4	fixed	705	1	10
gr_hk5	gr_hk5	fixed	705	1	10
gr_prsity3	gr_prsity3	fixed	705	1	1.2
gr_prsity4	gr_prsity4	fixed	705	1	1.2
gr_prsity5	gr_prsity5	fixed	705	1	1.2
gr_rech2	gr_rech2	fixed	705	1	1.1
gr_rech3	gr_rech3	fixed	705	1	1.1
...	...	...	...	...	...
gr_strt5	gr_strt5	fixed	705	1	1.05
gr_sy3	gr_sy3	fixed	705	1	1.75
gr_sy4	gr_sy4	fixed	705	1	1.75
gr_sy5	gr_sy5	fixed	705	1	1.75
gr_vka3	gr_vka3	fixed	705	1	10
gr_vka4	gr_vka4	fixed	705	1	10
gr_vka5	gr_vka5	fixed	705	1	10
pp_hk0	pp_hk0	log	32	0	1
pp_hk1	pp_hk1	log	32	0	1
pp_hk2	pp_hk2	log	32	0	1
pp_prsity0	pp_prsity0	log	32	0	0.0791812
pp_prsity1	pp_prsity1	log	32	0	0.0791812
pp_prsity2	pp_prsity2	log	32	0	0.0791812
pp_rech0	pp_rech0	log	32	0	0.0413927
pp_rech1	pp_rech1	fixed	32	1	1.1
pp_ss0	pp_ss0	log	32	0	1
pp_ss1	pp_ss1	log	32	0	1
pp_ss2	pp_ss2	log	32	0	1
pp_strt0	pp_strt0	fixed	32	1	1.05
pp_strt1	pp_strt1	fixed	32	1	1.05
pp_strt2	pp_strt2	fixed	32	1	1.05
pp_sy0	pp_sy0	log	32	0	0.243038

pp_sy1	pp_sy1	log	32	0	0.243038
pp_sy2	pp_sy2	log	32	0	0.243038
pp_vka0	pp_vka0	fixed	32	1	10
pp_vka1	pp_vka1	log	32	0	1
pp_vka2	pp_vka2	fixed	32	1	10
strk	strk	log	40	0	2
welflux	welflux	log	2	0	1
welflux_k02	welflux_k02	log	6	0	1

	lower bound	standard deviation
cn_hk6	-1	0.5
cn_hk7	-1	0.5
cn_hk8	-1	0.5
cn_prsity6	-0.09691	0.0440228
cn_prsity7	-0.09691	0.0440228
cn_prsity8	-0.09691	0.0440228
cn_rech4	-0.0457575	0.0217875
cn_rech5	-0.0457575	0.0217875
cn_ss6	-1	0.5
cn_ss7	-1	0.5
cn_ss8	-1	0.5
cn_strt6	-0.0222764	0.0108664
cn_strt7	-0.0222764	0.0108664
cn_strt8	-0.0222764	0.0108664
cn_sy6	-0.60206	0.211275
cn_sy7	-0.60206	0.211275
cn_sy8	-0.60206	0.211275
cn_vka6	-1	0.5
cn_vka7	-1	0.5
cn_vka8	-1	0.5
drncond_k00	-1	0.5
flow	-0.124939	0.0554622
gr_hk3	0.1	2.475
gr_hk4	0.1	2.475
gr_hk5	0.1	2.475
gr_prsity3	0.8	0.1
gr_prsity4	0.8	0.1
gr_prsity5	0.8	0.1
gr_rech2	0.9	0.05
gr_rech3	0.9	0.05
...	...	...
gr_strt5	0.95	0.025
gr_sy3	0.25	0.375
gr_sy4	0.25	0.375
gr_sy5	0.25	0.375
gr_vka3	0.1	2.475
gr_vka4	0.1	2.475
gr_vka5	0.1	2.475

```
pp_hk0              -1            0.5
pp_hk1              -1            0.5
pp_hk2              -1            0.5
pp_prsity0    -0.09691      0.0440228
pp_prsity1    -0.09691      0.0440228
pp_prsity2    -0.09691      0.0440228
pp_rech0    -0.0457575      0.0217875
pp_rech1           0.9           0.05
pp_ss0              -1            0.5
pp_ss1              -1            0.5
pp_ss2              -1            0.5
pp_strt0          0.95          0.025
pp_strt1          0.95          0.025
pp_strt2          0.95          0.025
pp_sy0        -0.60206       0.211275
pp_sy1        -0.60206       0.211275
pp_sy2        -0.60206       0.211275
pp_vka0            0.1          2.475
pp_vka1             -1            0.5
pp_vka2            0.1          2.475
strk                -2              1
welflux             -1            0.5
welflux_k02         -1            0.5

[65 rows x 7 columns]
```

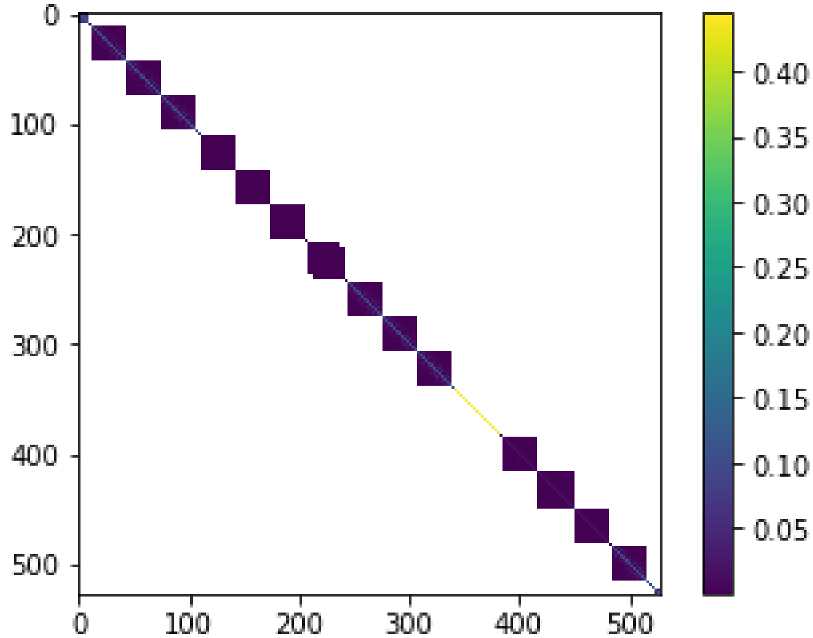
first ingredient: parameter covariance matrix (representing prior uncertainty in this instance)

In [4]:
```
cov = pyemu.Cov.from_binary(os.path.join(m_d,"prior_cov.jcb")).to_dataframe()
cov = cov.loc[pst.adj_par_names,pst.adj_par_names]
cov = pyemu.Cov.from_dataframe(cov)
```

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

In [5]:
```
# let's inspect only
x = cov.x.copy()
x[x<1e-7] = np.nan
c = plt.imshow(x)
plt.colorbar()
```

Out[5]:
```
<matplotlib.colorbar.Colorbar at 0x1b50090c470>
```




```
In [6]: pst.adj_par_groups
```

```
Out[6]: ['cn_hk6',
         'cn_hk7',
         'cn_hk8',
         'cn_prsity6',
         'cn_prsity7',
         'cn_prsity8',
         'cn_rech4',
         'cn_rech5',
         'cn_ss6',
         'cn_ss7',
         'cn_ss8',
         'cn_strt6',
         'cn_strt7',
         'cn_strt8',
         'cn_sy6',
         'cn_sy7',
         'cn_sy8',
         'cn_vka6',
         'cn_vka7',
         'cn_vka8',
         'drncond_k00',
         'flow',
         'pp_hk0',
         'pp_hk1',
```

```
 'pp_hk2',
 'pp_prsity0',
 'pp_prsity1',
 'pp_prsity2',
 'pp_rech0',
 'pp_ss0',
 'pp_ss1',
 'pp_ss2',
 'pp_sy0',
 'pp_sy1',
 'pp_sy2',
 'pp_vka1',
 'strk',
 'welflux',
 'welflux_k02']
```

second ingredient: jacobian matrix

```
In [7]: jco = os.path.join(m_d,"freyberg_pp.jcb")
```

the third ingredient--the (diagonal) noise covariance matrix--populated on-the-fly using weights when constructing the Schur object below...

```
In [8]: sc = pyemu.Schur(jco=jco,parcov=cov)
```

```
In [9]: sc
```

```
Out[9]: <pyemu.sc.Schur at 0x1b50092f940>
```

1.0.1 there we have it--all computations done and contained within sc. We will only be required to access different parts of sc below...

1.0.2 Parameter uncertainty

First let's inspect the (approx) posterior parameter covariance matrix and the reduction in parameter uncertainty through "data assimilation", before mapping to forecasts... (note that this matrix is *not* forecast-specific)

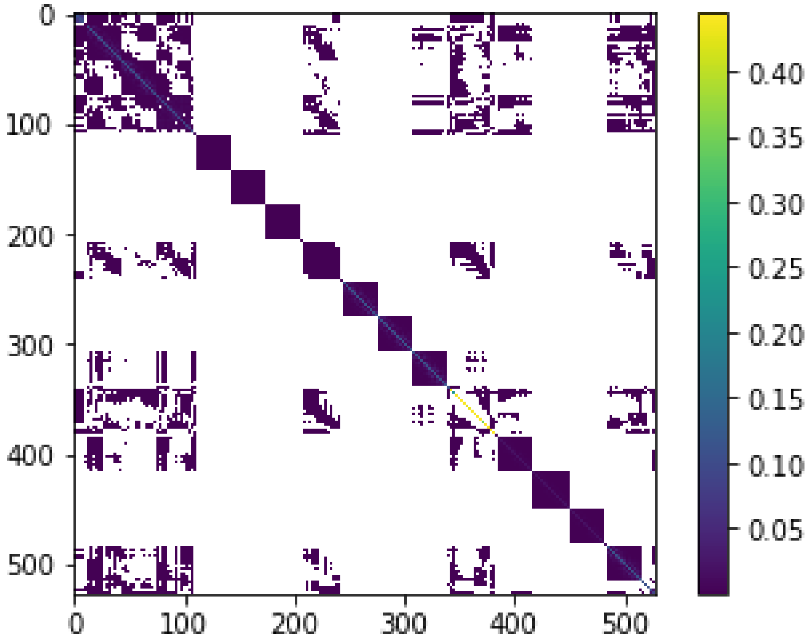
```
In [10]: sc.posterior_parameter.to_dataframe().sort_index(axis=1).iloc[100:105:,100:105]
```

Out[10]:

	hk225	hk226	hk227	hk228	hk229
hk225	0.095214	0.021767	0.003569	0.001025	0.020754
hk226	0.021767	0.084505	0.021604	0.006714	0.029579
hk227	0.003569	0.021604	0.093146	0.031621	0.012542
hk228	0.001025	0.006714	0.031621	0.105268	0.002141
hk229	0.020754	0.029579	0.012542	0.002141	0.084591

```
In [11]: x = sc.posterior_parameter.x.copy()
         x[x<1e-7] = np.nan
         c = plt.imshow(x)
         plt.colorbar(c)
```

```
Out[11]: <matplotlib.colorbar.Colorbar at 0x1b5012ce9e8>
```



We can see the posterior variance for each parameter along the diagonal. The off-diags are symmetric.

```
In [12]: par_sum = sc.get_parameter_summary().sort_values("percent_reduction",ascending=False)
         par_sum
```

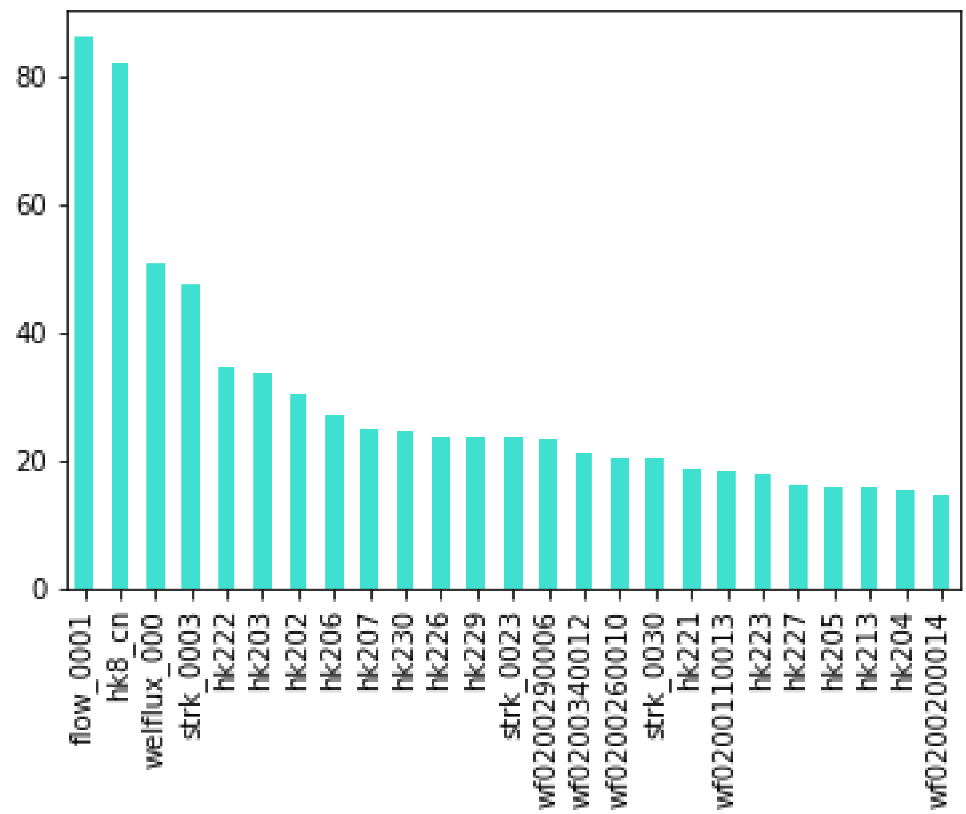
Out[12]:

	percent_reduction	post_var	prior_var
flow_0001	8.629753e+01	0.000187	0.001367
hk8_cn	8.216237e+01	0.019820	0.111111
welflux_000	5.076704e+01	0.054703	0.111111
strk_0003	4.767839e+01	0.232540	0.444444
hk222	3.457443e+01	0.072695	0.111111
hk203	3.372133e+01	0.073643	0.111111
hk202	3.028185e+01	0.077465	0.111111
hk206	2.695334e+01	0.081163	0.111111
hk207	2.519217e+01	0.083120	0.111111
hk230	2.464115e+01	0.083732	0.111111
hk226	2.394589e+01	0.084505	0.111111
hk229	2.386831e+01	0.084591	0.111111
strk_0023	2.368509e+01	0.339177	0.444444
wf0200290006	2.340886e+01	0.085101	0.111111
wf0200340012	2.139508e+01	0.087339	0.111111
wf0200260010	2.061345e+01	0.088207	0.111111
strk_0030	2.044844e+01	0.353562	0.444444
hk221	1.895753e+01	0.090047	0.111111

```
wf0200110013       1.846858e+01  0.090590   0.111111
hk223              1.796930e+01  0.091145   0.111111
hk227              1.616835e+01  0.093146   0.111111
hk205              1.602062e+01  0.093310   0.111111
hk213              1.577955e+01  0.093578   0.111111
hk204              1.551881e+01  0.093868   0.111111
wf0200200014       1.474544e+01  0.094727   0.111111
hk6_cn             1.447790e+01  0.095025   0.111111
hk210              1.440933e+01  0.095101   0.111111
hk225              1.430781e+01  0.095214   0.111111
hk220              1.367623e+01  0.095915   0.111111
wf0200090016       1.353431e+01  0.096073   0.111111
...                         ...       ...        ...
ss119             -2.220446e-14  0.111111   0.111111
rech5_cn          -2.220446e-14  0.000211   0.000211
ss000             -2.220446e-14  0.111111   0.111111
ss123             -2.220446e-14  0.111111   0.111111
sy203             -2.220446e-14  0.019839   0.019839
sy213             -2.220446e-14  0.019839   0.019839
ss002             -2.220446e-14  0.111111   0.111111
ss012             -2.220446e-14  0.111111   0.111111
ss115             -2.220446e-14  0.111111   0.111111
ss007             -2.220446e-14  0.111111   0.111111
ss008             -2.220446e-14  0.111111   0.111111
ss113             -2.220446e-14  0.111111   0.111111
ss114             -2.220446e-14  0.111111   0.111111
sy210             -4.440892e-14  0.019839   0.019839
prsity009         -4.440892e-14  0.000861   0.000861
sy110             -4.440892e-14  0.019839   0.019839
sy220             -4.440892e-14  0.019839   0.019839
prsity209         -4.440892e-14  0.000861   0.000861
sy116             -4.440892e-14  0.019839   0.019839
prsity025         -4.440892e-14  0.000861   0.000861
prsity215         -4.440892e-14  0.000861   0.000861
prsity115         -4.440892e-14  0.000861   0.000861
sy120             -4.440892e-14  0.019839   0.019839
sy216             -4.440892e-14  0.019839   0.019839
prsity015         -4.440892e-14  0.000861   0.000861
prsity225         -4.440892e-14  0.000861   0.000861
prsity125         -4.440892e-14  0.000861   0.000861
prsity000         -6.661338e-14  0.000861   0.000861
ss102             -6.661338e-14  0.111111   0.111111
prsity200         -6.661338e-14  0.000861   0.000861

[527 rows x 3 columns]
```

In [13]:
```
par_sum.loc[par_sum.index[:25],"percent_reduction"].plot(kind="bar",color="turquoise")
```

Out[13]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1b5006c1c18>
```



What have we achieved by "notionally calibrating" our model to 13 head and 1 stream flow observations? Which parameters are informed? Will they matter for the forecast of interest? Which ones are un-informed?

```
In [14]: pst.nnz_obs_names
```

```
Out[14]: ['fo_39_19791230',
          'hds_00_002_009_000',
          'hds_00_002_015_000',
          'hds_00_003_008_000',
          'hds_00_009_001_000',
          'hds_00_013_010_000',
          'hds_00_015_016_000',
          'hds_00_021_010_000',
          'hds_00_022_015_000',
          'hds_00_024_004_000',
          'hds_00_026_006_000',
          'hds_00_029_015_000',
          'hds_00_033_007_000',
          'hds_00_034_010_000']
```

1.1 Forecast uncertainty

In [15]:
```
forecasts = sc.pst.pestpp_options['forecasts'].split(",")
forecasts
```

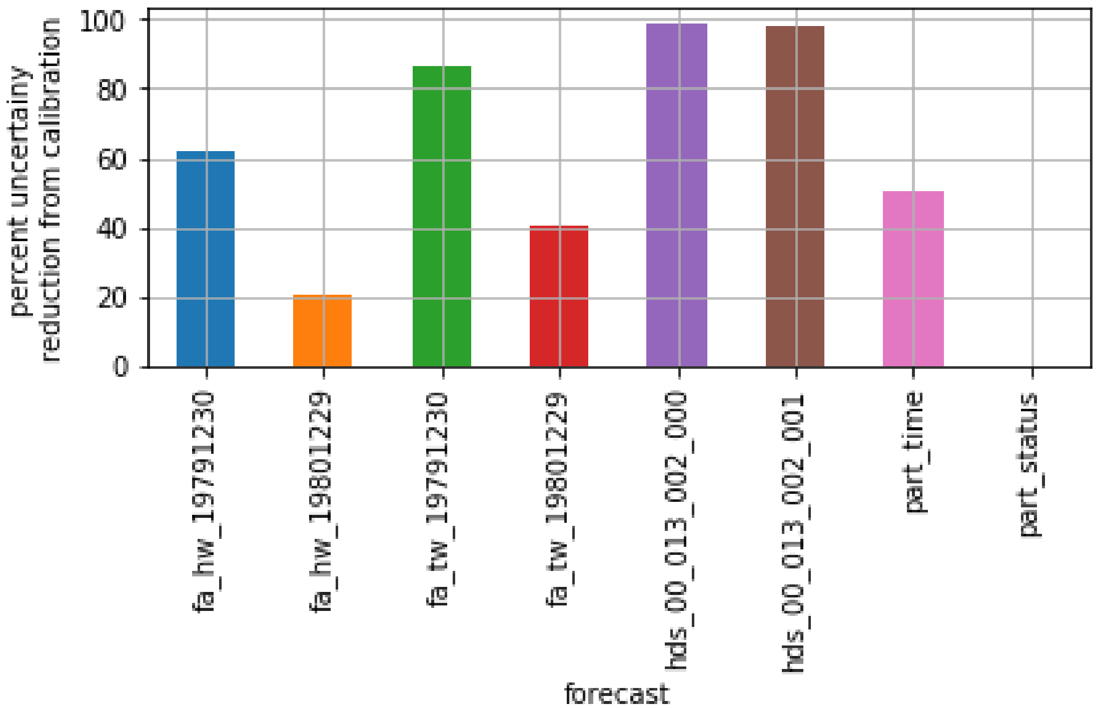
Out[15]:
```
['fa_hw_19791230',
 'fa_hw_19801229',
 'fa_tw_19791230',
 'fa_tw_19801229',
 'hds_00_013_002_000',
 'hds_00_013_002_001',
 'part_time',
 'part_status']
```

In [16]:
```
df = sc.get_forecast_summary()
df
```

Out[16]:

	percent_reduction	post_var	prior_var
fa_hw_19791230	62.250126	48708.787021	129030.329448
fa_hw_19801229	20.164045	277258.059336	347284.702910
fa_tw_19791230	86.866072	22609.793588	172147.988428
fa_tw_19801229	40.797559	220463.794436	372389.706408
hds_00_013_002_000	98.537703	0.111608	7.632373
hds_00_013_002_001	97.727870	0.192769	8.484073
part_time	50.756746	93226.847888	189319.023019
part_status	NaN	0.000000	0.000000

In [17]:
```
# make a pretty plot
fig = plt.figure()
ax = plt.subplot(111)
ax = df["percent_reduction"].plot(kind='bar',ax=ax,grid=True)
ax.set_ylabel("percent uncertainy\nreduction from calibration")
ax.set_xlabel("forecast")
plt.tight_layout()
```



Surprise, surprise... Some forecasts benefit from calibration, some do not!

1.1.1 Before moving onto data worth, let's look at the contribution of different parameters to forecast uncertainty

Parameter contributions to uncertainty are quantified by "fixing" parameters (or parameter groups) and observing the uncertainty reduction as a result. This approach is of course subject to some sizable assumptions--related to parameter representativeness. But it can be very informative. Let's do by group.

In [18]:
```
par_contrib = sc.get_par_group_contribution()
```

In [19]:
```
par_contrib.head()
```

Out[19]:

	fa_hw_19791230	fa_hw_19801229	fa_tw_19791230	fa_tw_19801229
base	48708.787021	277258.059336	22609.793588	220463.794436
cn_hk6	48343.538434	277079.164728	22509.835083	220406.965378
cn_hk7	48708.780545	277258.057052	22609.791510	220463.730874
cn_hk8	44990.748125	275607.303800	21321.626080	220162.729424
cn_prsity6	48708.787021	277258.059336	22609.793588	220463.794436

	hds_00_013_002_000	hds_00_013_002_001	part_status	part_time
base	0.111608	0.192769	0.0	93226.847888
cn_hk6	0.110868	0.192766	0.0	87722.159943
cn_hk7	0.111608	0.192769	0.0	93226.236725

cn_hk8	0.108634	0.187237	0.0	89722.098031
cn_prsity6	0.111608	0.192769	0.0	93166.415889

In [20]:
```
base = par_contrib.loc["base",:]
par_contrib = 100.0 * (base - par_contrib) / par_contrib
par_contrib.sort_index()
```

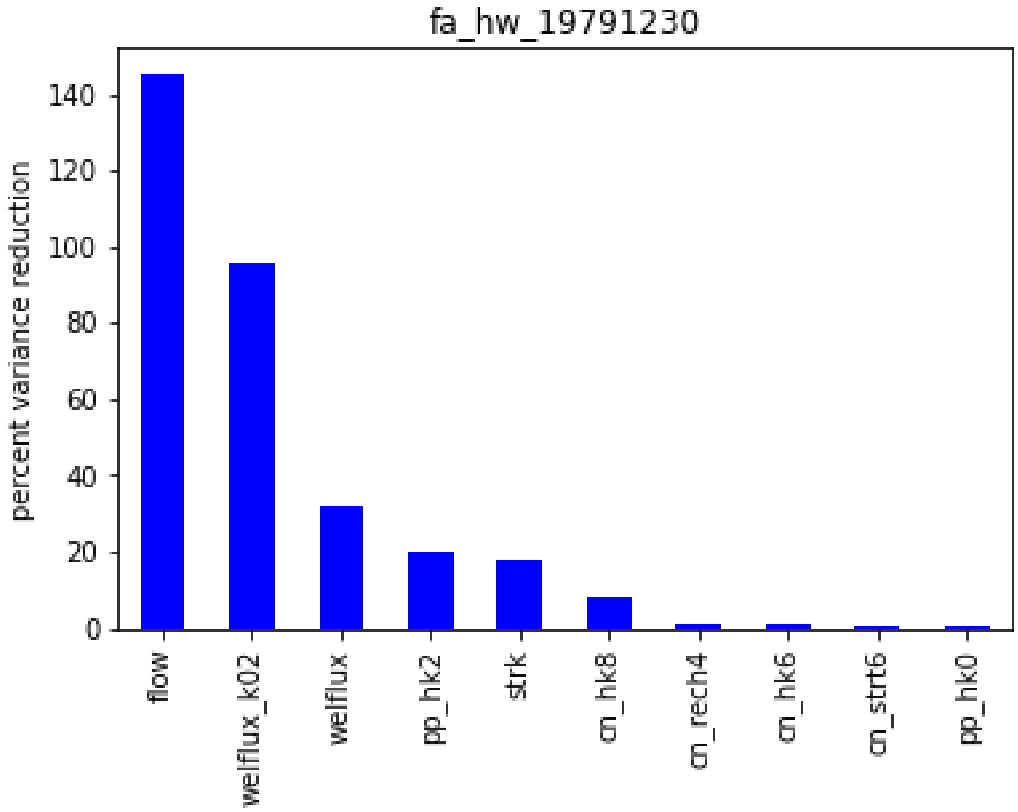
Out[20]:

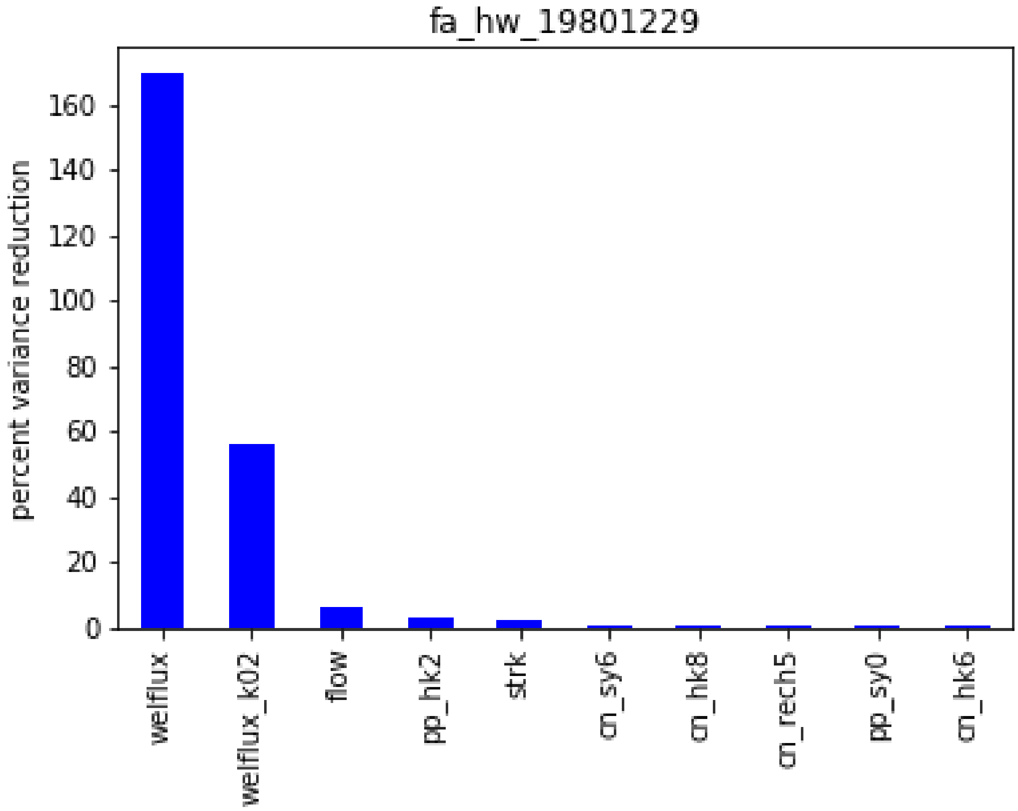
	fa_hw_19791230	fa_hw_19801229	fa_tw_19791230	fa_tw_19801229	\
base	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	
cn_hk6	7.555272e-01	6.456444e-02	4.440659e-01	2.578369e-02	
cn_hk7	1.329528e-05	8.237482e-07	9.192837e-06	2.883103e-05	
cn_hk8	8.264008e+00	5.989520e-01	6.041601e+00	1.367466e-01	
cn_prsity6	-1.493767e-14	0.000000e+00	-4.827084e-14	-1.320118e-14	
cn_prsity7	-1.493767e-14	0.000000e+00	-4.827084e-14	-1.320118e-14	
cn_prsity8	-1.493767e-14	0.000000e+00	-4.827084e-14	-1.320118e-14	
cn_rech4	1.328983e+00	2.071189e-03	4.946222e-01	5.626596e-02	
cn_rech5	0.000000e+00	1.523076e-01	-6.436111e-14	1.485863e-01	
cn_ss6	2.789907e-06	2.104129e-05	3.491816e-05	4.579829e-05	
cn_ss7	0.000000e+00	1.341250e-05	-8.045139e-14	4.648073e-05	
cn_ss8	4.617341e-06	1.873997e-03	1.033730e-05	3.116114e-03	
cn_strt6	7.331869e-01	3.392103e-04	2.381097e-01	2.695676e-02	
cn_strt7	5.529236e-06	3.978286e-09	2.567813e-06	1.265512e-07	
cn_strt8	2.419169e-04	4.321156e-06	1.585604e-04	1.990229e-05	
cn_sy6	6.185704e-03	6.450432e-01	6.508401e-04	7.492785e-01	
cn_sy7	-1.493767e-14	2.099404e-14	-9.654167e-14	0.000000e+00	
cn_sy8	-1.493767e-14	2.099404e-14	-9.654167e-14	0.000000e+00	
cn_vka6	9.106484e-04	1.018885e-04	8.569676e-04	1.135336e-05	
cn_vka7	5.049578e-02	4.055652e-03	9.290548e-03	7.240419e-04	
cn_vka8	6.929210e-02	6.021175e-03	1.129875e-02	1.091725e-03	
drncond_k00	7.415039e-04	2.742403e-06	3.318787e-03	1.220112e-05	
flow	1.450607e+02	6.080339e+00	1.071328e+02	6.934249e-01	
pp_hk0	4.369761e-01	5.119383e-02	4.359639e-01	2.238615e-02	
pp_hk1	8.760160e-04	2.577004e-05	4.482011e-04	1.659310e-05	
pp_hk2	2.033927e+01	2.805611e+00	6.856688e+01	3.596352e+00	
pp_prsity0	0.000000e+00	4.198807e-14	0.000000e+00	1.320118e-14	
pp_prsity1	0.000000e+00	4.198807e-14	0.000000e+00	1.320118e-14	
pp_prsity2	0.000000e+00	4.198807e-14	0.000000e+00	1.320118e-14	
pp_rech0	3.796292e-01	1.300183e-02	1.487279e-01	1.560503e-02	
pp_ss0	0.000000e+00	5.870941e-08	0.000000e+00	3.408468e-06	
pp_ss1	0.000000e+00	2.146002e-08	0.000000e+00	2.411191e-05	
pp_ss2	8.469699e-07	4.384063e-04	8.631079e-07	9.606797e-04	
pp_sy0	1.615596e-03	1.026042e-01	5.105185e-04	1.415935e-01	
pp_sy1	1.493767e-14	0.000000e+00	-1.609028e-14	1.320118e-14	
pp_sy2	1.493767e-14	0.000000e+00	-1.609028e-14	1.320118e-14	
pp_vka1	1.641495e-02	1.580767e-03	1.411553e-02	5.243871e-04	
strk	1.809544e+01	1.847978e+00	3.201253e+01	1.023446e+00	
welflux	3.198862e+01	1.691162e+02	1.998213e+01	3.606368e+02	
welflux_k02	9.593429e+01	5.615419e+01	1.701436e+01	3.050788e+01	

	hds_00_013_002_000	hds_00_013_002_001	part_status	part_time
base	0.000000e+00	0.000000e+00	NaN	0.000000e+00
cn_hk6	6.671577e-01	1.871559e-03	NaN	6.275140e+00
cn_hk7	2.262146e-05	1.334577e-05	NaN	6.555697e-04
cn_hk8	2.737385e+00	2.954897e+00	NaN	3.906228e+00
cn_prsity6	-3.481633e-13	-1.295851e-13	NaN	6.486457e-02
cn_prsity7	-3.481633e-13	-1.295851e-13	NaN	1.079401e-02
cn_prsity8	-3.481633e-13	-1.295851e-13	NaN	2.835530e+00
cn_rech4	2.382578e-03	1.769951e+00	NaN	1.091228e+00
cn_rech5	-1.616473e-13	1.226362e+00	NaN	0.000000e+00
cn_ss6	2.229798e-06	1.687027e-03	NaN	2.478430e-07
cn_ss7	-4.227697e-13	4.042413e-04	NaN	2.896643e-07
cn_ss8	2.360833e-09	1.958794e-02	NaN	1.669725e-05
cn_strt6	2.320961e-03	1.037450e+00	NaN	6.400388e-01
cn_strt7	9.819126e-09	7.336089e-06	NaN	7.730557e-04
cn_strt8	3.924209e-06	3.123582e-04	NaN	6.912381e-03
cn_sy6	5.964960e-04	1.134598e+01	NaN	1.550628e-02
cn_sy7	-3.481633e-13	-1.007884e-13	NaN	-1.560915e-14
cn_sy8	-3.481633e-13	-1.007884e-13	NaN	-1.560915e-14
cn_vka6	8.860983e-05	1.026596e-06	NaN	7.298465e-06
cn_vka7	8.102339e-04	7.150283e-04	NaN	5.039768e-04
cn_vka8	4.389541e-04	7.316762e-04	NaN	3.857006e-04
drncond_k00	2.470427e-04	8.354980e-04	NaN	1.347509e-04
flow	8.728178e-01	2.013697e+00	NaN	6.478075e-01
pp_hk0	4.180533e+00	2.567447e+00	NaN	4.611903e+00
pp_hk1	7.707142e-04	2.290446e-03	NaN	5.029761e-04
pp_hk2	1.155114e+02	6.540933e+01	NaN	5.266906e+02
pp_prsity0	-2.486881e-14	1.295851e-13	NaN	5.408669e-02
pp_prsity1	-2.486881e-14	1.295851e-13	NaN	7.196968e-03
pp_prsity2	-2.486881e-14	1.295851e-13	NaN	9.585220e-01
pp_rech0	8.684877e-02	3.435903e-01	NaN	2.434414e-01
pp_ss0	-2.486881e-14	2.129983e-04	NaN	8.360607e-07
pp_ss1	-2.486881e-14	4.319504e-14	NaN	1.340384e-07
pp_ss2	2.149449e-06	2.412443e-03	NaN	5.339847e-06
pp_sy0	1.893636e-03	2.307920e+00	NaN	4.711510e-03
pp_sy1	1.243440e-13	5.759339e-14	NaN	1.560915e-14
pp_sy2	1.243440e-13	5.759339e-14	NaN	1.560915e-14
pp_vka1	8.755905e-04	1.056924e-03	NaN	3.190442e-04
strk	3.077665e-01	1.780119e-01	NaN	2.757848e-01
welflux	5.857188e-01	1.879207e+01	NaN	2.361491e-01
welflux_k02	2.538238e-01	1.984225e+00	NaN	1.258342e+00

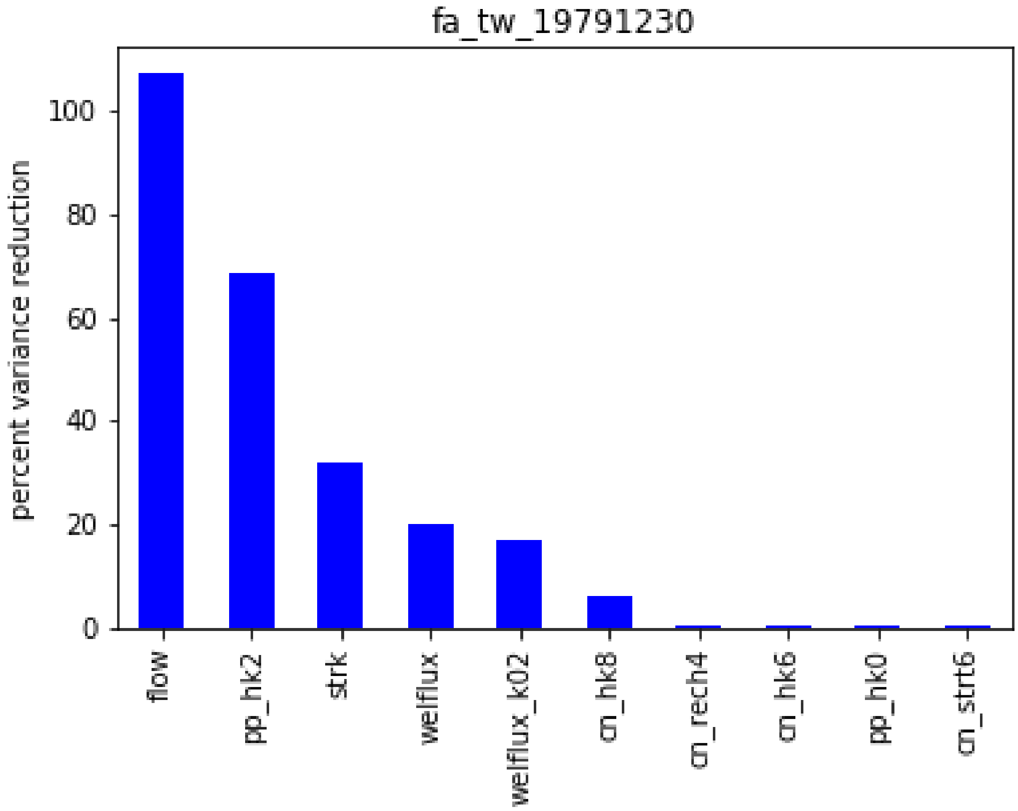
```
In [21]: for forecast in par_contrib.columns:
             fore_df = par_contrib.loc[:,forecast].copy()
             fore_df.sort_values(inplace=True, ascending=False)
             ax = fore_df.iloc[:10].plot(kind="bar",color="b")
             ax.set_title(forecast)
```

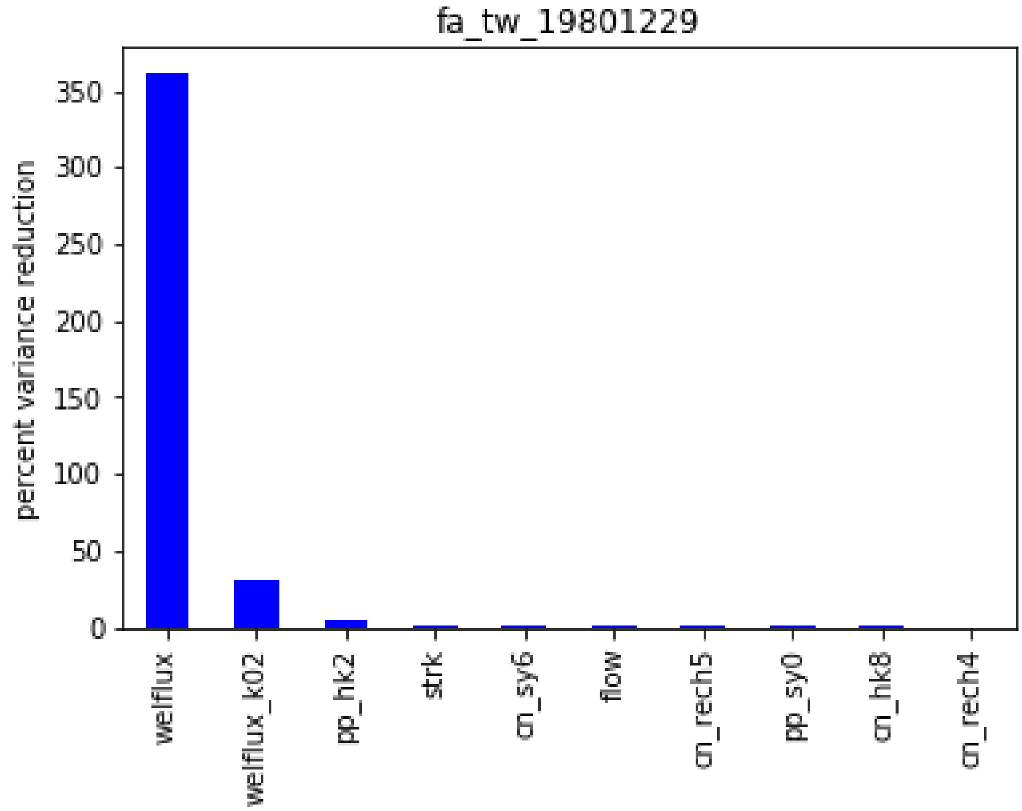
```
ax.set_ylabel("percent variance reduction")
plt.show()
```

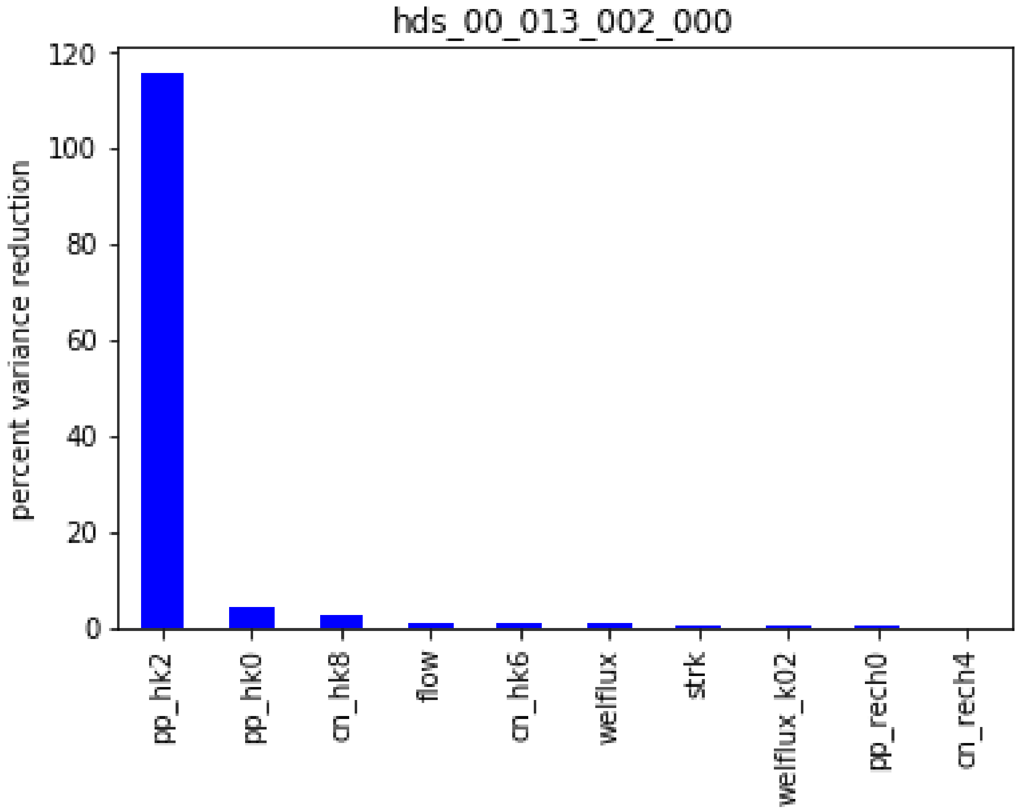


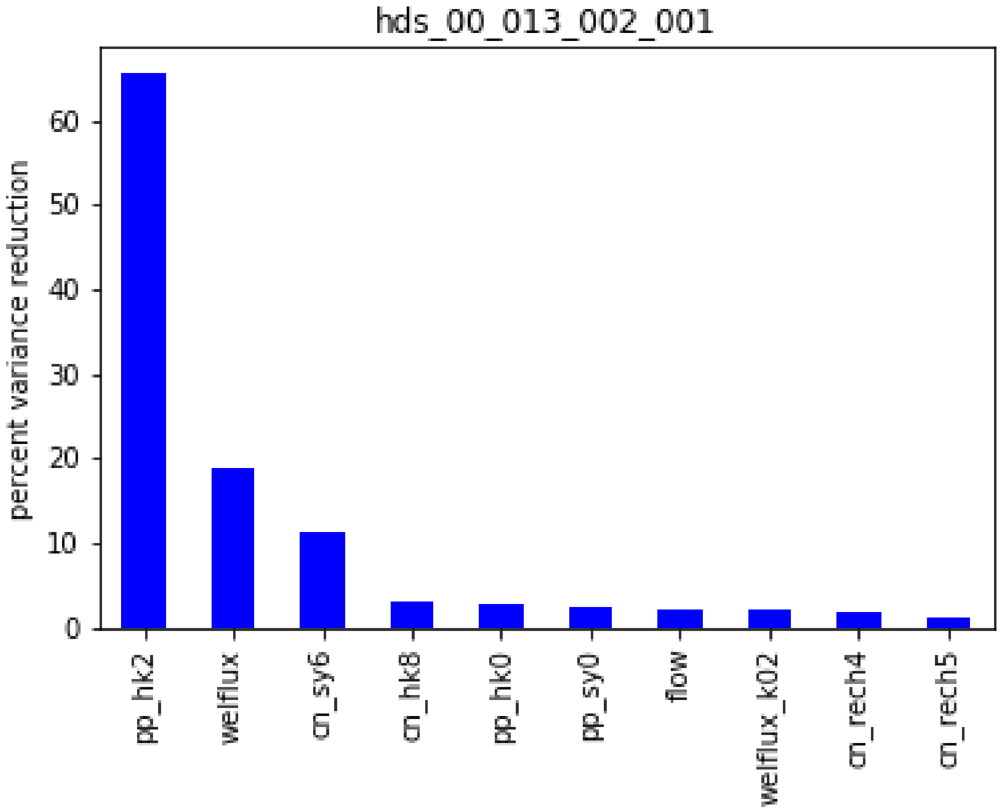


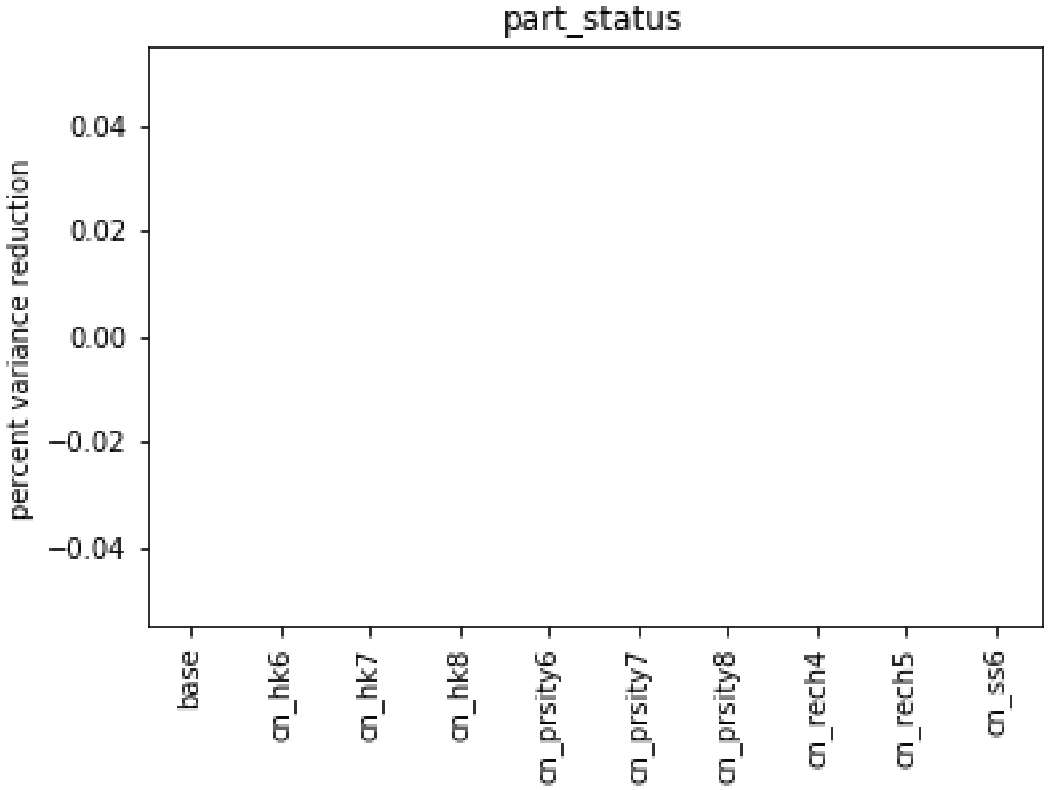
fa_hw_19801229
percent variance reduction
0
20
40
60
80
100
120
140
160
welflux
welflux_k02
flow
pp_hk2
strk
cn_sy6
cn_hk8
cn_rech5
pp_sy0
cn_hk6

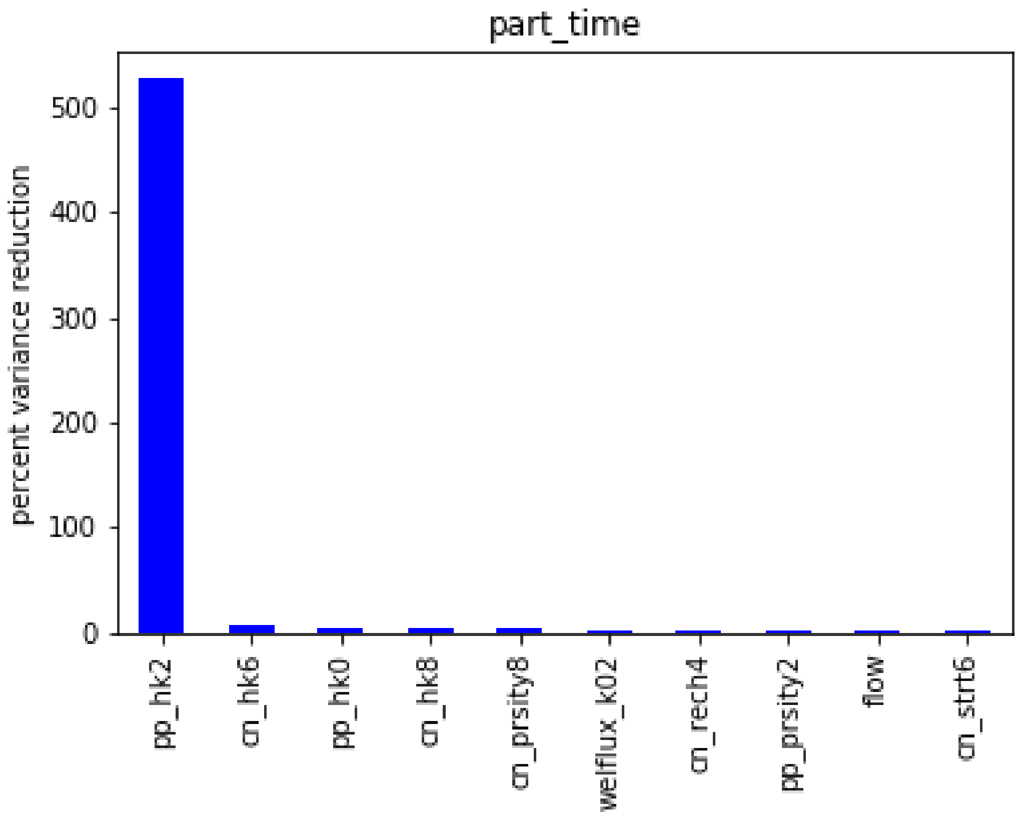

fa_tw_19791230
percent variance reduction
100
80
60
40
20
0
flow
pp_hk2
strk
welflux
welflux_k02
cn_hk8
cn_rech4
cn_hk6
pp_hk0
cn_strt6


fa_tw_19801229
percent variance reduction
0
50
100
150
200
250
300
350
welflux
welflux_k02
pp_hk2
strk
cn_sy6
flow
cn_rech5
pp_sy0
cn_hk8
cn_rech4


hds_00_013_002_000
percent variance reduction
0
20
40
60
80
100
120
pp_hk2
pp_hk0
cn_hk8
flow
cn_hk6
welflux
strk
welflux_k02
pp_rech0
cn_rech4


hds_00_013_002_001
percent variance reduction
0
10
20
30
40
50
60
pp_hk2
welflux
cn_sy6
cn_hk8
pp_hk0
pp_sy0
flow
welflux_k02
cn_rech4
cn_rech5


part_status
percent variance reduction
0.04
0.02
0.00
−0.02
−0.04
base
cn_hk6
cn_hk7
cn_hk8
cn_prsity6
cn_prsity7
cn_prsity8
cn_rech4
cn_rech5
cn_ss6



1.1.2 Data worth

1.1.3 what is the worth of *existing* observations?

What is happening under the hood is that we are recalculating the Schur complement without some of the observations to see how the posterior forecast uncertainty increases (wrt a "base" condition in which we have all observation data available).

In [22]:
```
dw_rm = sc.get_removed_obs_importance()
dw_rm.head()
```

Out[22]:

	fa_hw_19791230	fa_hw_19801229	fa_tw_19791230	\
base	48708.787021	277258.059336	22609.793588	
fo_39_19791230	51191.377140	278273.504686	23178.615771	
hds_00_002_009_000	48940.420250	277387.232706	22615.754525	
hds_00_002_015_000	48743.445105	277373.413424	22616.617273	
hds_00_003_008_000	48880.074463	277344.626609	22610.161149	

	fa_tw_19801229	hds_00_013_002_000	hds_00_013_002_001	\
base	220463.794436	0.111608	0.192769	
fo_39_19791230	220465.341686	0.111749	0.193247	
hds_00_002_009_000	220464.858302	0.111696	0.192831	

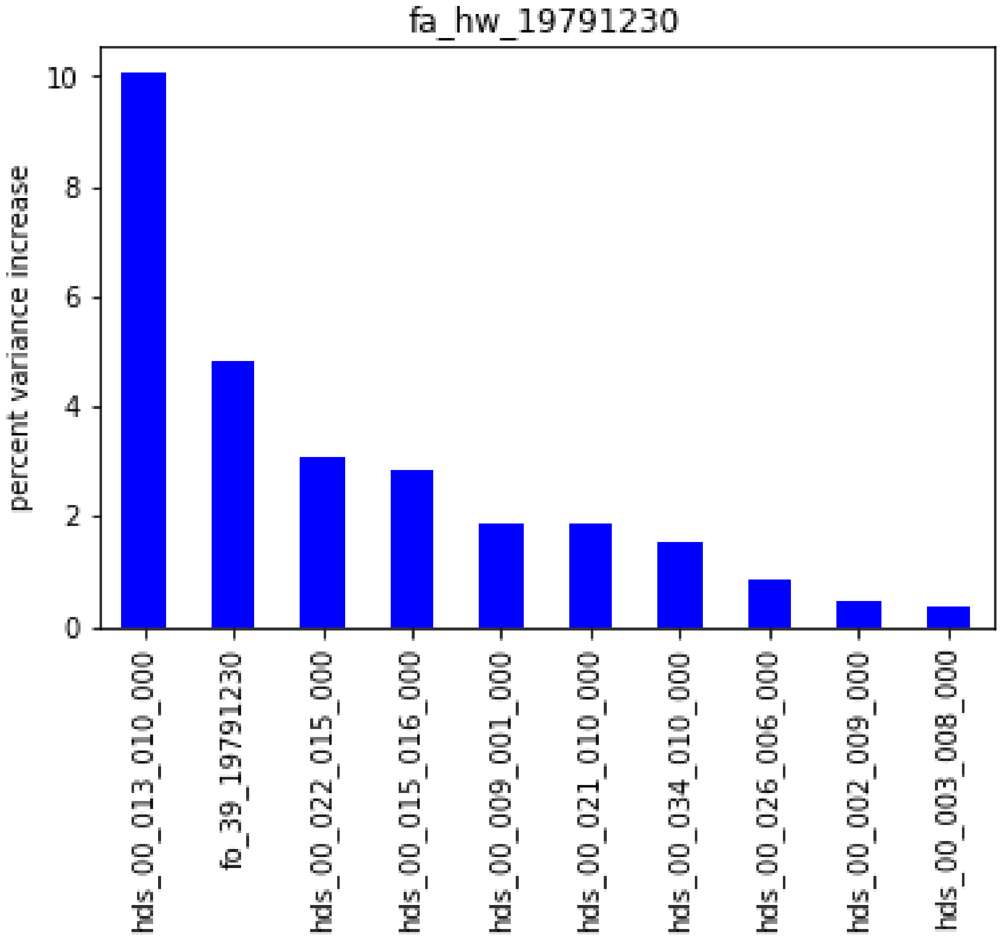
```
hds_00_002_015_000   220464.501038          0.111613          0.192769
hds_00_003_008_000   220463.889982          0.111804          0.192926

                    part_status     part_time
base                        0.0  93226.847888
fo_39_19791230              0.0  93298.497985
hds_00_002_009_000          0.0  93388.870852
hds_00_002_015_000          0.0  93253.399795
hds_00_003_008_000          0.0  94604.086151
```

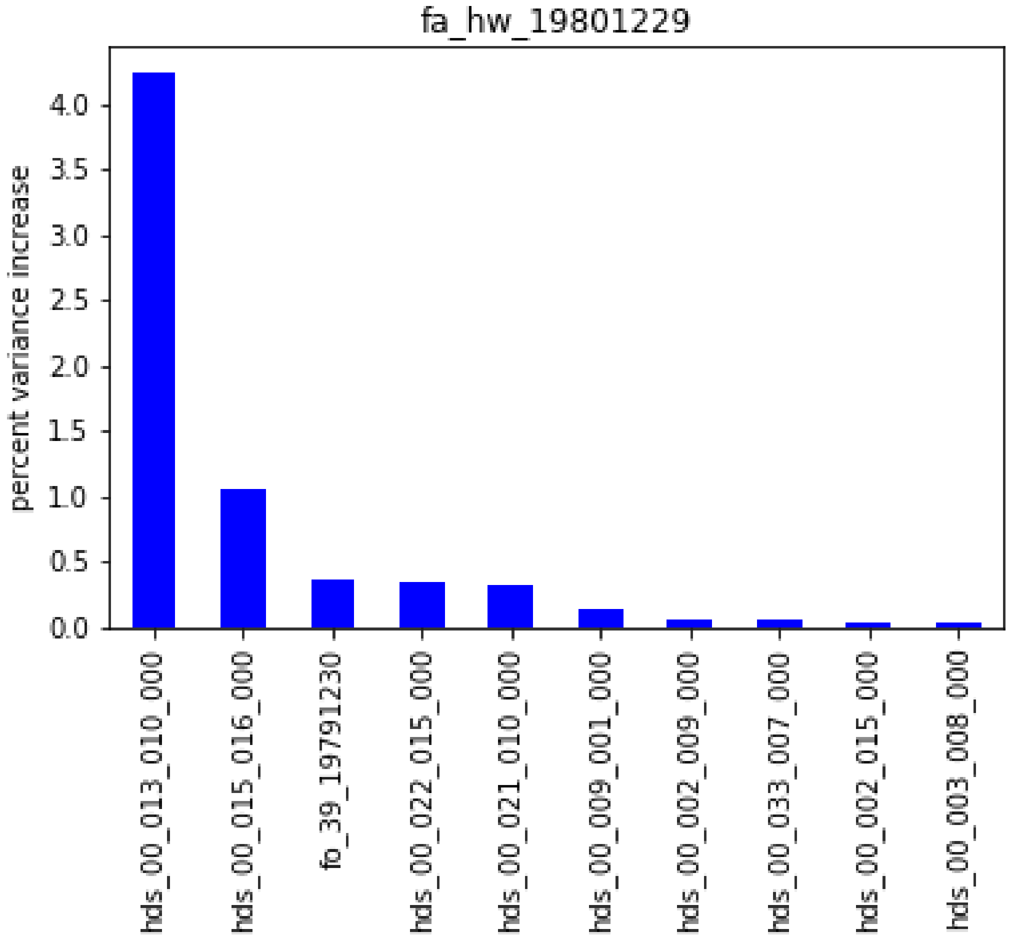
Here the base row contains the results of the Schur complement calculation (in terms of forecast uncertainty variance) using all observations.

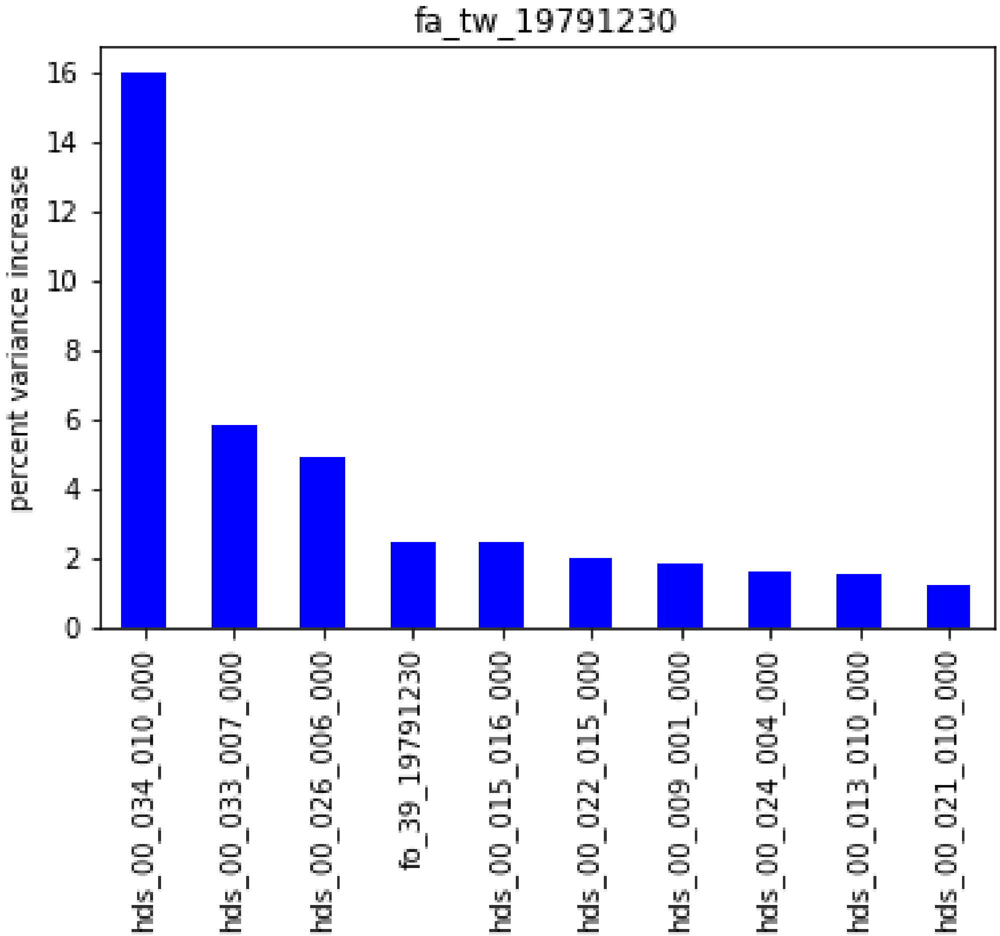
```
In [23]: # let's normalize to make more meaningful comparisons of data worth (unctainty varian
         base = dw_rm.loc["base",:]
         dw_rm = 100 * (dw_rm  - base) / dw_rm
         dw_rm.head()
```

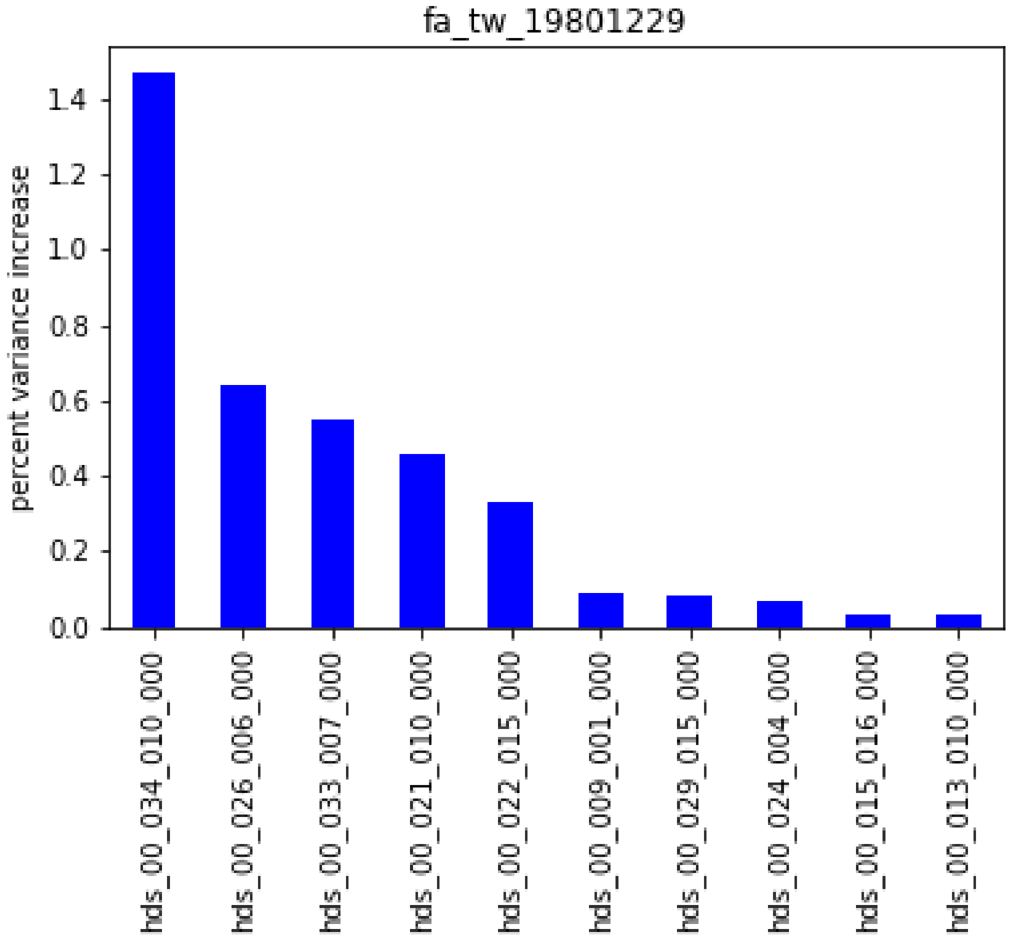
```
Out[23]:                    fa_hw_19791230  fa_hw_19801229  fa_tw_19791230  \
         base                     0.000000        0.000000        0.000000
         fo_39_19791230           4.849626        0.364909        2.454082
         hds_00_002_009_000       0.473296        0.046568        0.026357
         hds_00_002_015_000       0.071103        0.041588        0.030171
         hds_00_003_008_000       0.350424        0.031213        0.001626

                            fa_tw_19801229  hds_00_013_002_000  hds_00_013_002_001  \
         base                     0.000000            0.000000            0.000000
         fo_39_19791230           0.000702            0.126100            0.247304
         hds_00_002_009_000       0.000483            0.078499            0.031874
         hds_00_002_015_000       0.000321            0.004694            0.000123
         hds_00_003_008_000       0.000043            0.175625            0.081108

                            part_status  part_time
         base                       NaN   0.000000
         fo_39_19791230             NaN   0.076797
         hds_00_002_009_000         NaN   0.173493
         hds_00_002_015_000         NaN   0.028473
         hds_00_003_008_000         NaN   1.455792
```

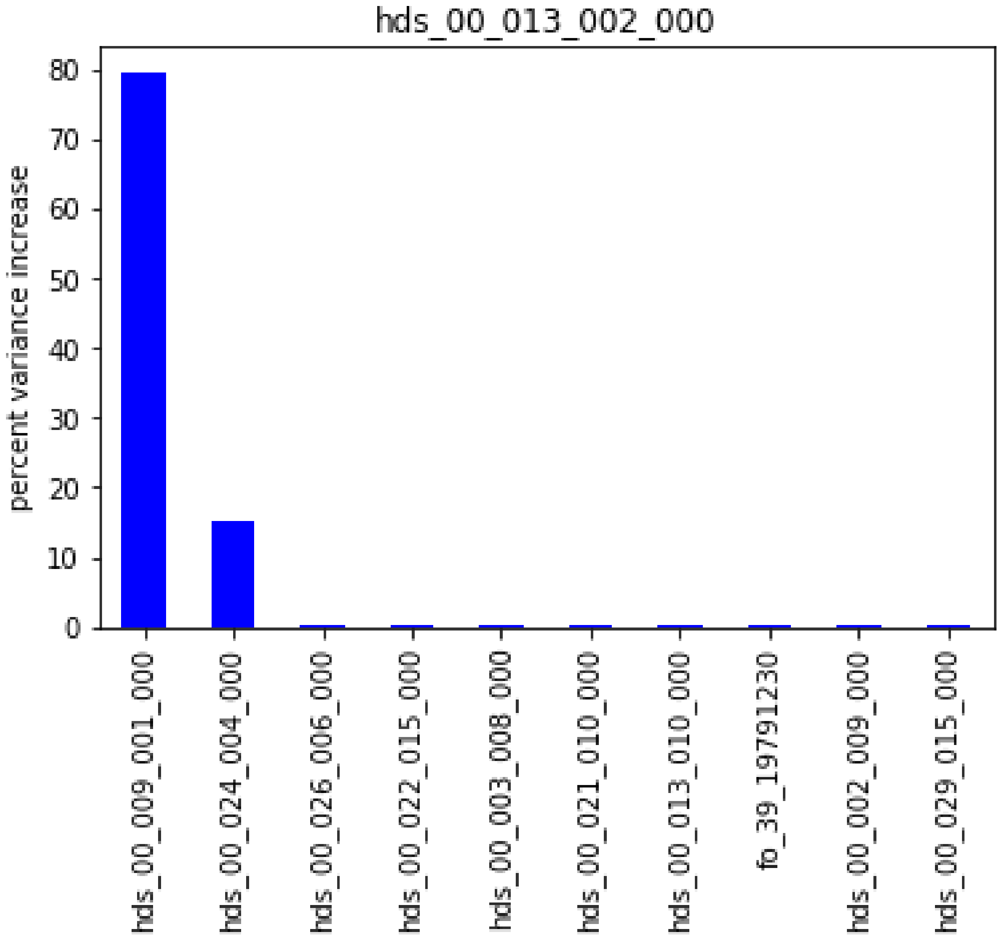
```
In [24]: for forecast in dw_rm.columns:
             fore_df = dw_rm.loc[:,forecast].copy()
             fore_df.sort_values(inplace=True, ascending=False)
             ax = fore_df.iloc[:10].plot(kind="bar",color="b")
             ax.set_title(forecast)
             ax.set_ylabel("percent variance increase")
             plt.show()
```

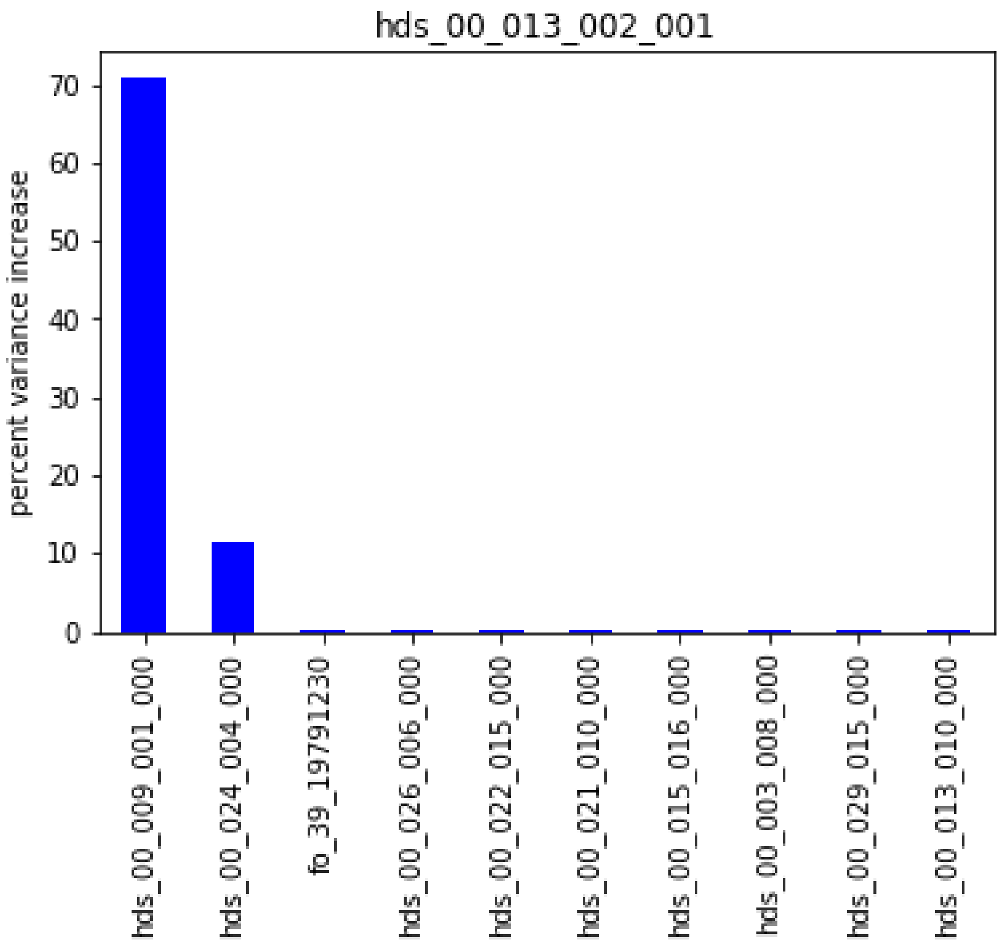

fa_hw_19791230
percent variance increase
10
8
6
4
2
0
hds_00_013_010_000
fo_39_19791230
hds_00_022_015_000
hds_00_015_016_000
hds_00_009_001_000
hds_00_021_010_000
hds_00_034_010_000
hds_00_026_006_000
hds_00_002_009_000
hds_00_003_008_000

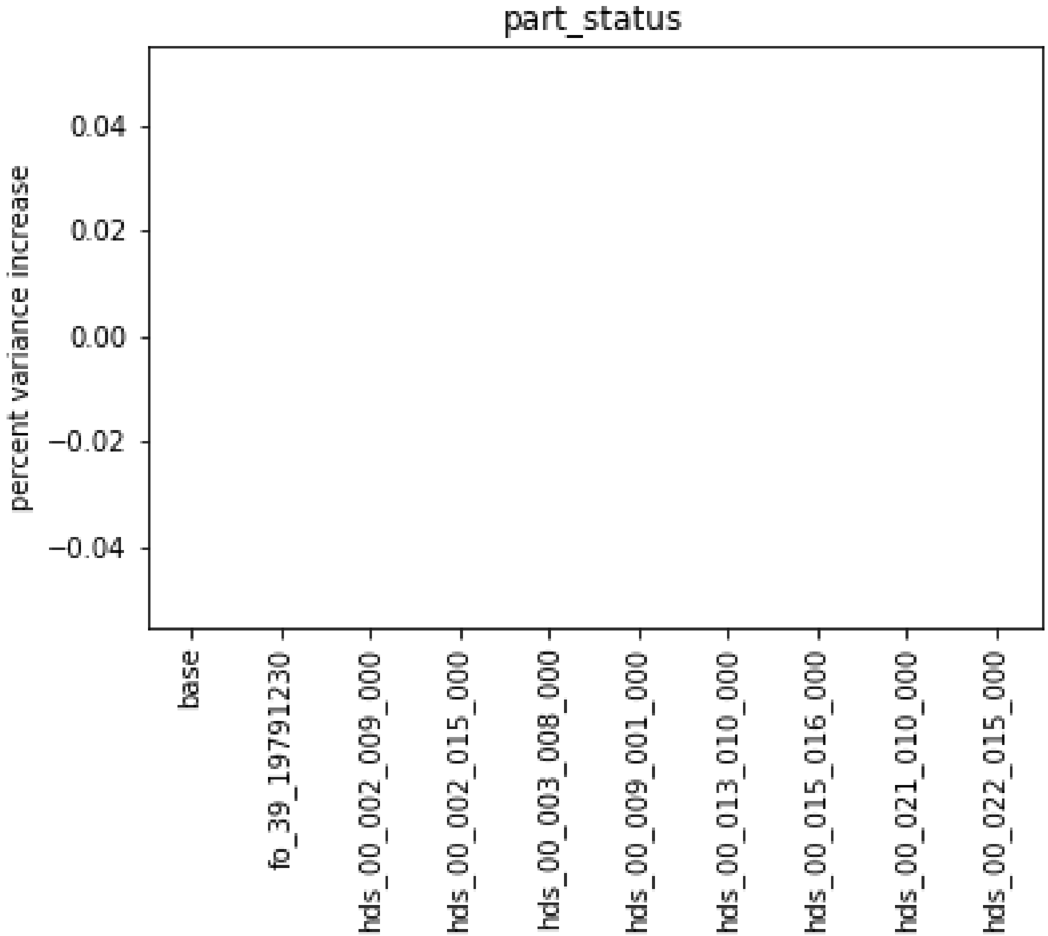

fa_hw_19801229
percent variance increase
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
hds_00_013_010_000
hds_00_015_016_000
fo_39_19791230
hds_00_022_015_000
hds_00_021_010_000
hds_00_009_001_000
hds_00_002_009_000
hds_00_033_007_000
hds_00_002_015_000
hds_00_003_008_000

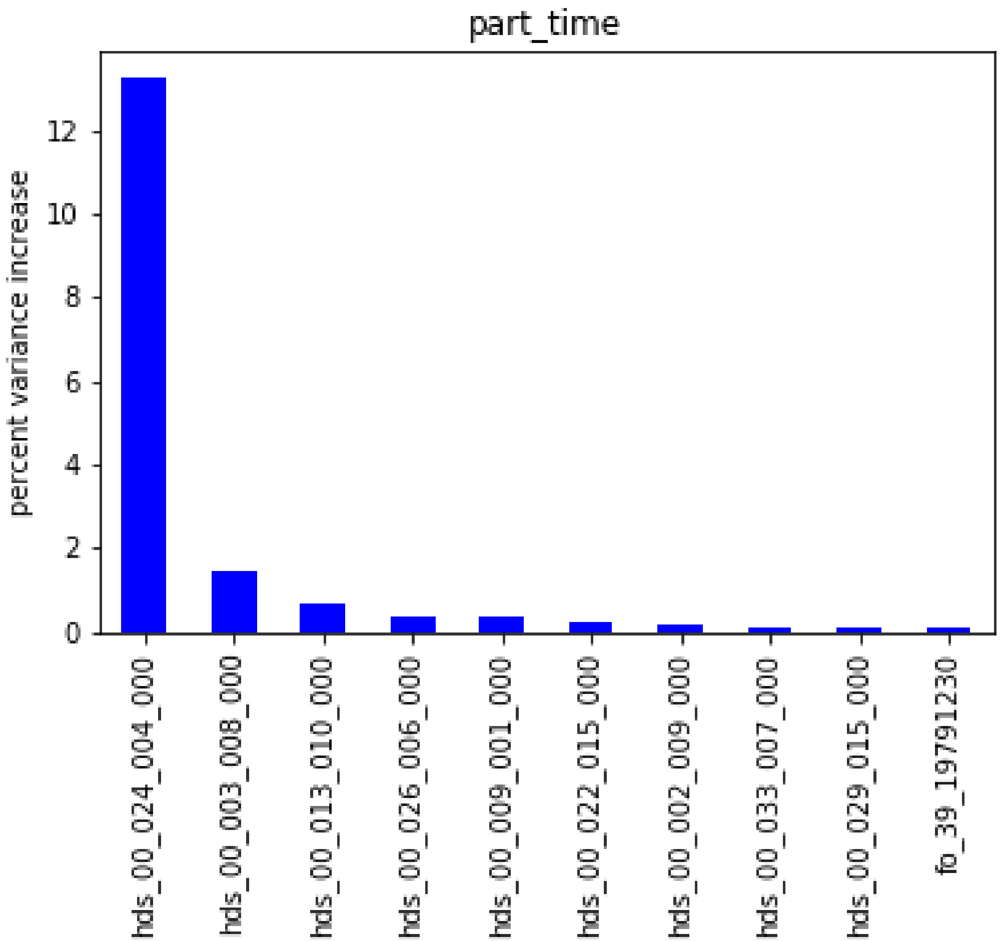

fa_tw_19791230
percent variance increase
16
14
12
10
8
6
4
2
0
hds_00_034_010_000
hds_00_033_007_000
hds_00_026_006_000
fo_39_19791230
hds_00_015_016_000
hds_00_022_015_000
hds_00_009_001_000
hds_00_024_004_000
hds_00_013_010_000
hds_00_021_010_000


fa_tw_19801229
percent variance increase
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
hds_00_034_010_000
hds_00_026_006_000
hds_00_033_007_000
hds_00_021_010_000
hds_00_022_015_000
hds_00_009_001_000
hds_00_029_015_000
hds_00_024_004_000
hds_00_015_016_000
hds_00_013_010_000


hds_00_013_002_000
percent variance increase
80
70
60
50
40
30
20
10
0
hds_00_009_001_000
hds_00_024_004_000
hds_00_026_006_000
hds_00_022_015_000
hds_00_003_008_000
hds_00_021_010_000
hds_00_013_010_000
fo_39_19791230
hds_00_002_009_000
hds_00_029_015_000

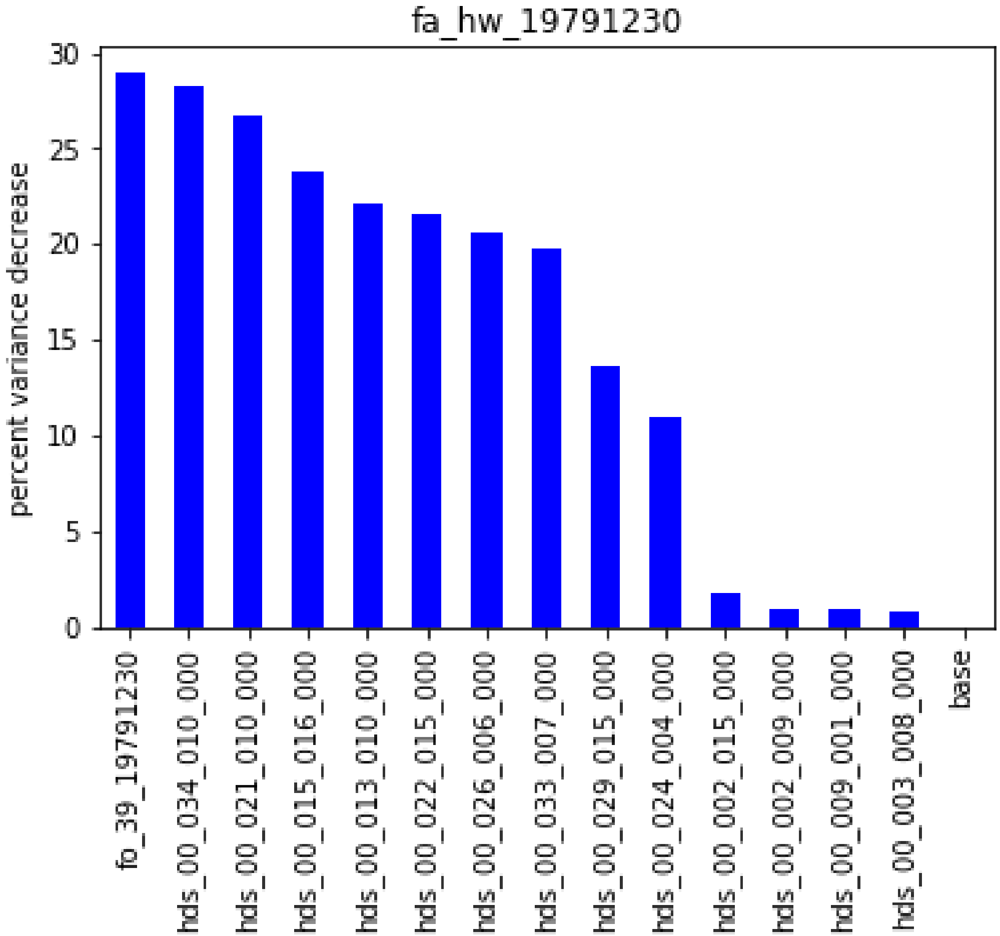

hds_00_013_002_001
percent variance increase
0
10
20
30
40
50
60
70
hds_00_009_001_000
hds_00_024_004_000
fo_39_19791230
hds_00_026_006_000
hds_00_022_015_000
hds_00_021_010_000
hds_00_015_016_000
hds_00_003_008_000
hds_00_029_015_000
hds_00_013_010_000

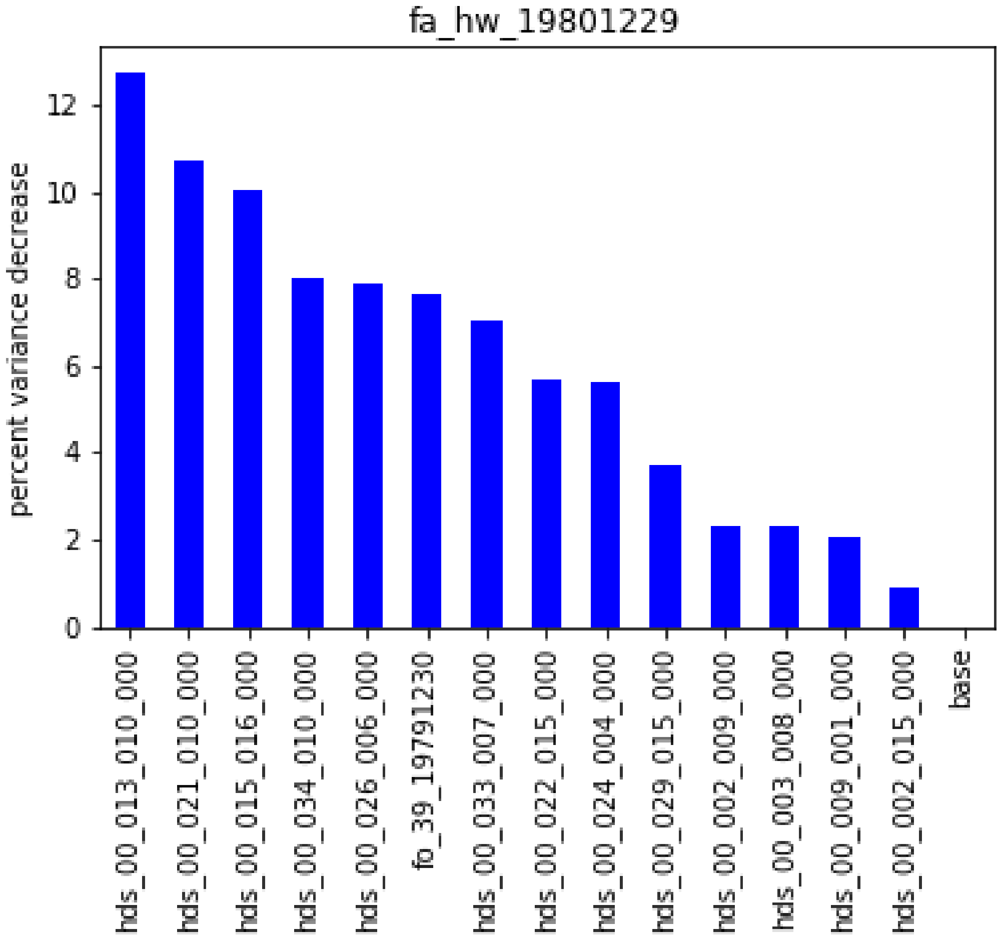

part_status
percent variance increase
0.04
0.02
0.00
−0.02
−0.04
base
fo_39_19791230
hds_00_002_009_000
hds_00_002_015_000
hds_00_003_008_000
hds_00_009_001_000
hds_00_013_010_000
hds_00_015_016_000
hds_00_021_010_000
hds_00_022_015_000

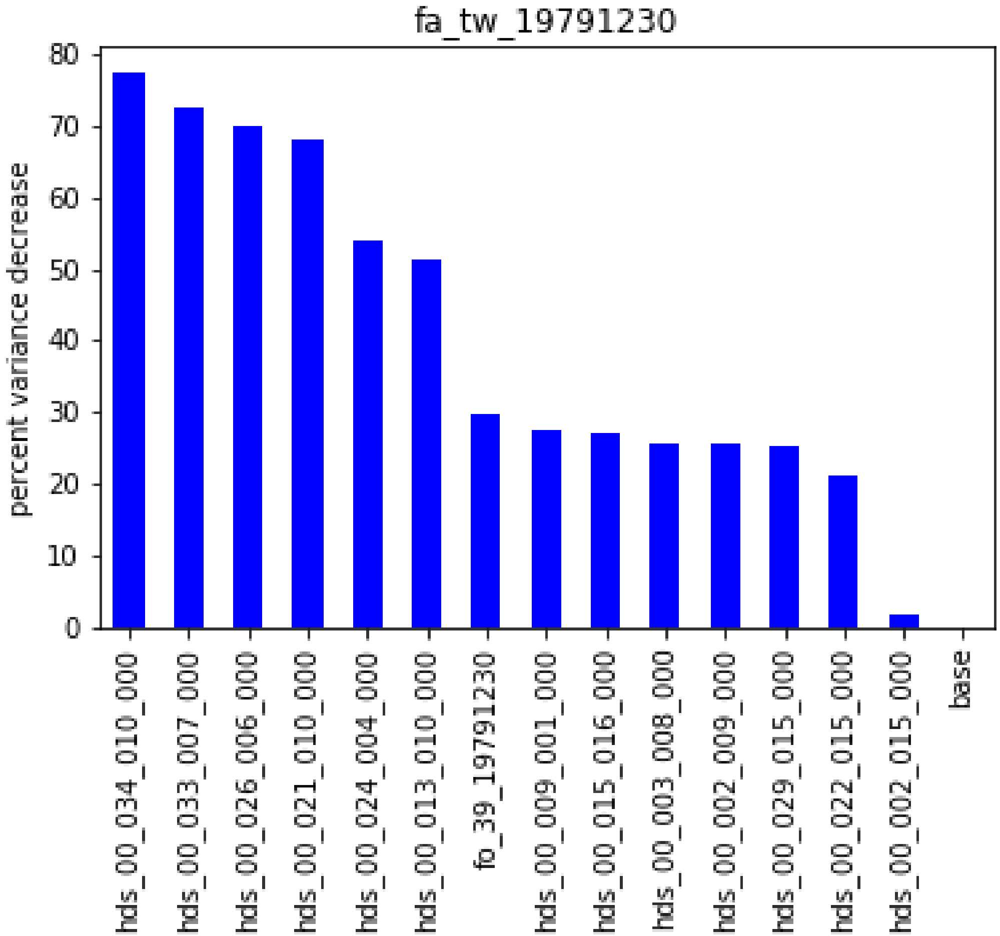


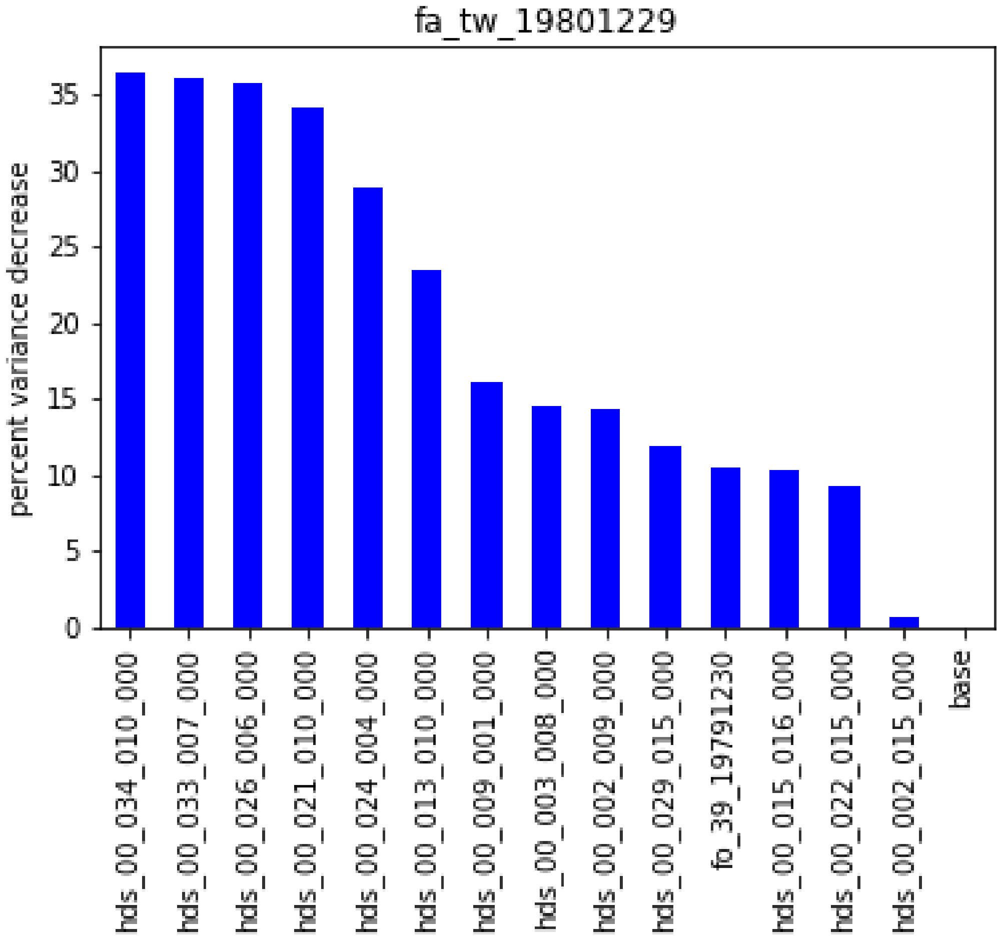
There is also an option to calculate the worth of observations by taking a "base" condition of zero observation (i.e., a priori) and calculating the reduction in uncertainty through adding observations to the dataset.

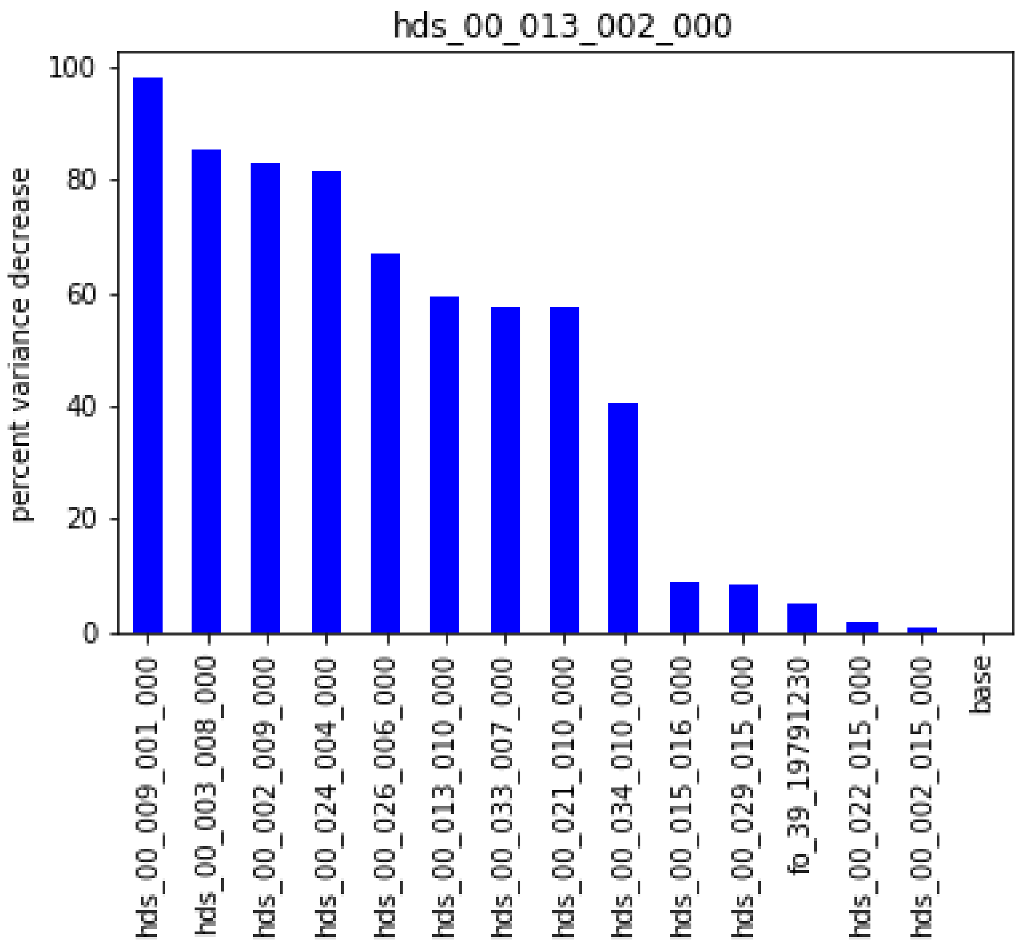
```
In [25]: dw_ad = sc.get_added_obs_importance()
         base = dw_ad.loc["base",:]
         dw_ad = 100 * (base - dw_ad) / base
         for forecast in dw_ad.columns:
             fore_df_ad = dw_ad.loc[:,forecast].copy()
             fore_df_ad.sort_values(inplace=True, ascending=False)
             ax = fore_df_ad.iloc[:20].plot(kind="bar",color="b")
             ax.set_title(forecast)
             ax.set_ylabel("percent variance decrease")
             plt.show()
```

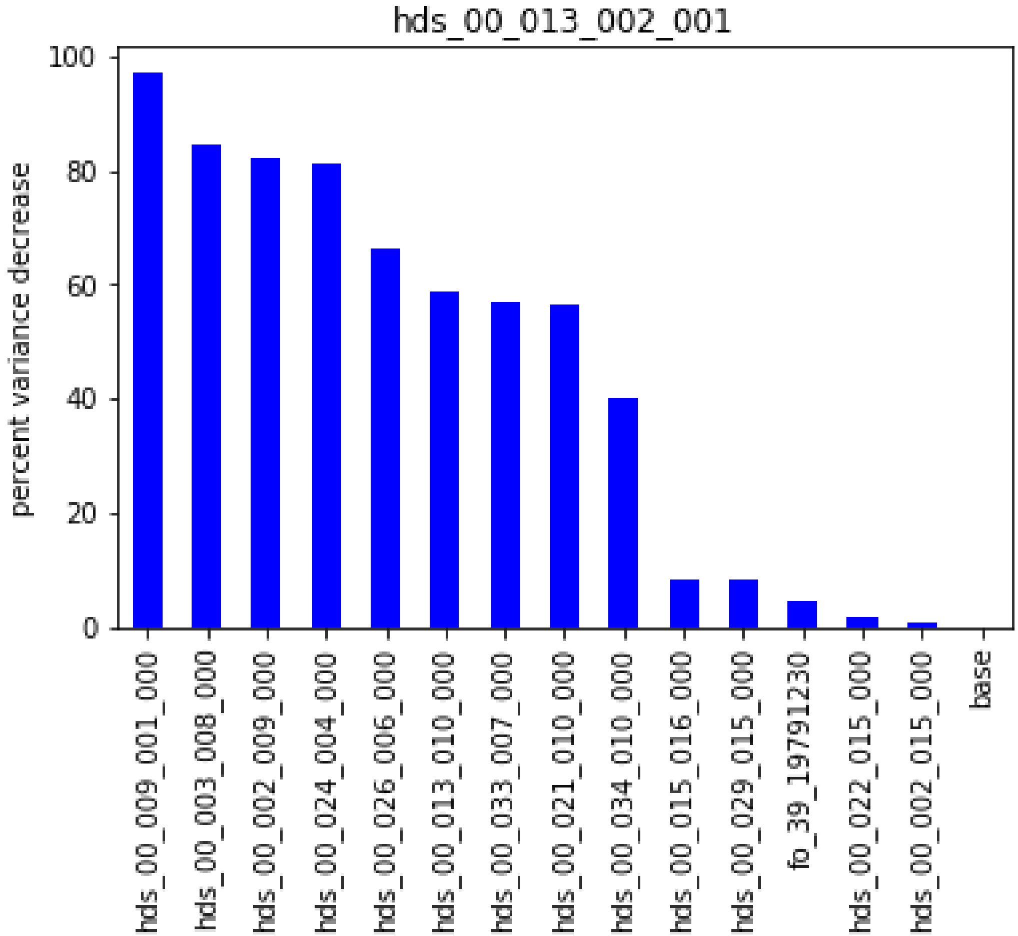

fa_hw_19791230
percent variance decrease
30
25
20
15
10
5
0
fo_39_19791230
hds_00_034_010_000
hds_00_021_010_000
hds_00_015_016_000
hds_00_013_010_000
hds_00_022_015_000
hds_00_026_006_000
hds_00_033_007_000
hds_00_029_015_000
hds_00_024_004_000
hds_00_002_015_000
hds_00_002_009_000
hds_00_009_001_000
hds_00_003_008_000
base

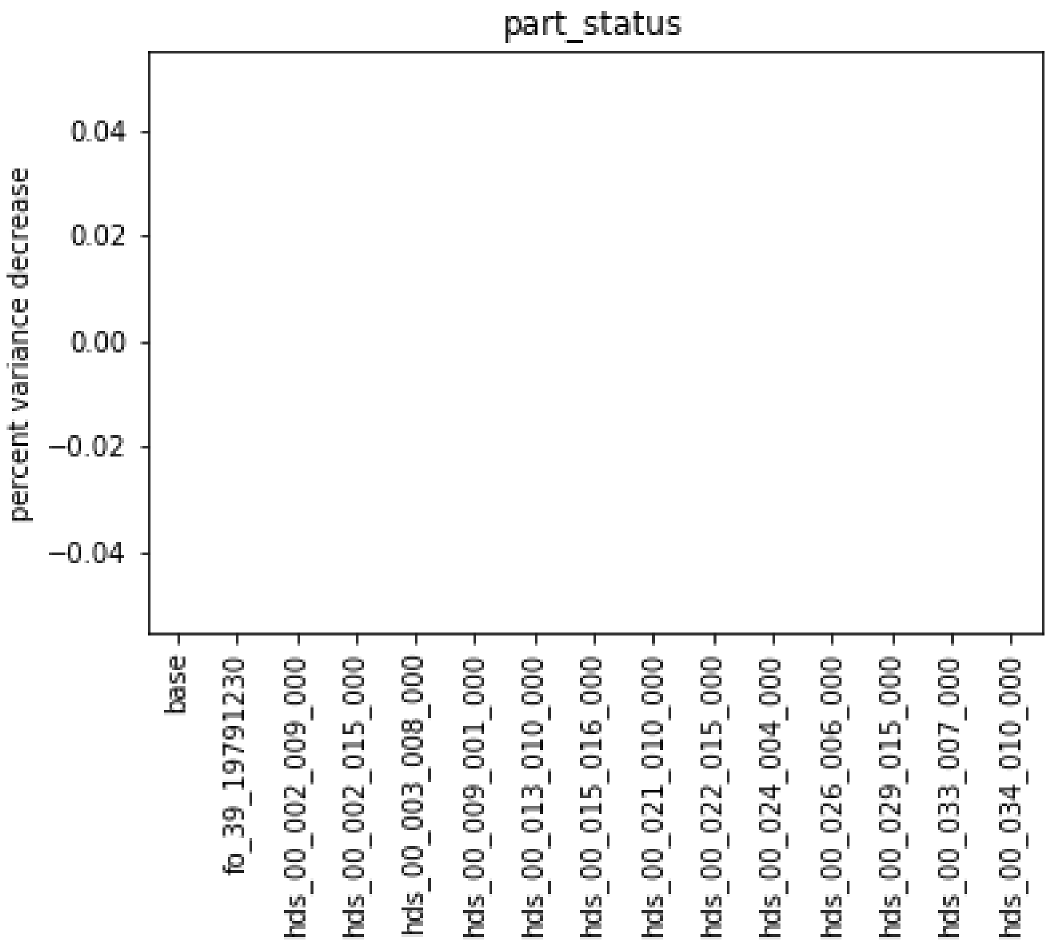

fa_hw_19801229
percent variance decrease
0
2
4
6
8
10
12
hds_00_013_010_000
hds_00_021_010_000
hds_00_015_016_000
hds_00_034_010_000
hds_00_026_006_000
fo_39_19791230
hds_00_033_007_000
hds_00_022_015_000
hds_00_024_004_000
hds_00_029_015_000
hds_00_002_009_000
hds_00_003_008_000
hds_00_009_001_000
hds_00_002_015_000
base

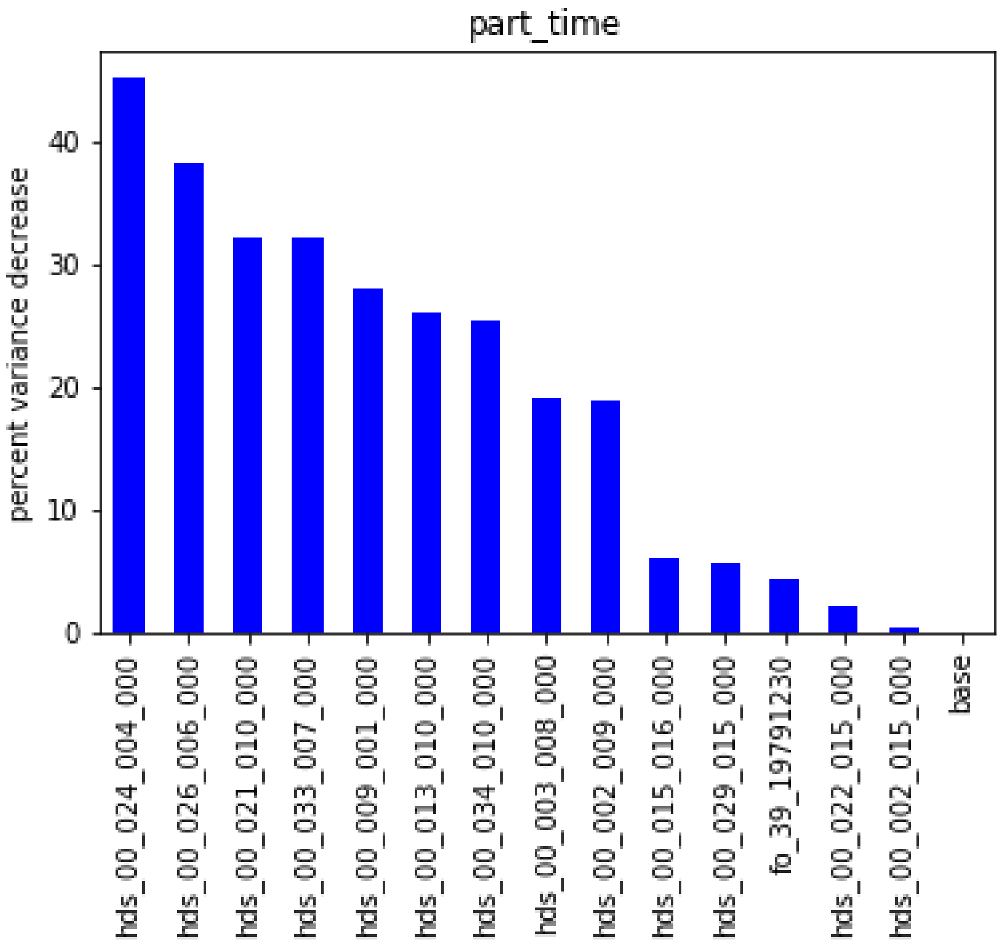

fa_tw_19791230
percent variance decrease
80
70
60
50
40
30
20
10
0
hds_00_034_010_000
hds_00_033_007_000
hds_00_026_006_000
hds_00_021_010_000
hds_00_024_004_000
hds_00_013_010_000
fo_39_19791230
hds_00_009_001_000
hds_00_015_016_000
hds_00_003_008_000
hds_00_002_009_000
hds_00_029_015_000
hds_00_022_015_000
hds_00_002_015_000
base


fa_tw_19801229
percent variance decrease
0
5
10
15
20
25
30
35
hds_00_034_010_000
hds_00_033_007_000
hds_00_026_006_000
hds_00_021_010_000
hds_00_024_004_000
hds_00_013_010_000
hds_00_009_001_000
hds_00_003_008_000
hds_00_002_009_000
hds_00_029_015_000
fo_39_19791230
hds_00_015_016_000
hds_00_022_015_000
hds_00_002_015_000
base


hds_00_013_002_000
percent variance decrease
0
20
40
60
80
100
hds_00_009_001_000
hds_00_003_008_000
hds_00_002_009_000
hds_00_024_004_000
hds_00_026_006_000
hds_00_013_010_000
hds_00_033_007_000
hds_00_021_010_000
hds_00_034_010_000
hds_00_015_016_000
hds_00_029_015_000
fo_39_19791230
hds_00_022_015_000
hds_00_002_015_000
base


hds_00_013_002_001
percent variance decrease
100
80
60
40
20
0
hds_00_009_001_000
hds_00_003_008_000
hds_00_002_009_000
hds_00_024_004_000
hds_00_026_006_000
hds_00_013_010_000
hds_00_033_007_000
hds_00_021_010_000
hds_00_034_010_000
hds_00_015_016_000
hds_00_029_015_000
fo_39_19791230
hds_00_022_015_000
hds_00_002_015_000
base


part_status
percent variance decrease
0.04
0.02
0.00
−0.02
−0.04
base
fo_39_19791230
hds_00_002_009_000
hds_00_002_015_000
hds_00_003_008_000
hds_00_009_001_000
hds_00_013_010_000
hds_00_015_016_000
hds_00_021_010_000
hds_00_022_015_000
hds_00_024_004_000
hds_00_026_006_000
hds_00_029_015_000
hds_00_033_007_000
hds_00_034_010_000



Do these two approaches give the same answer? They shouldn't.. Why? Let's discuss..

1.1.4 what is the worth of *potential* observations? what data should we collect?

Recall we are "carrying" cell-by-cell heads, reach-based sfr flows, etc..

In [26]:
```
z_obs = pst.observation_data.loc[(pst.observation_data.weight == 0),"obsnme"].tolist()
z_obs = [x for x in z_obs if x not in forecasts]  # less our forecasts
z_obs
```

Out[26]:
```
['fa_0_19791230',
 'fa_0_19801229',
 'fa_10_19791230',
 'fa_10_19801229',
 'fa_11_19791230',
 'fa_11_19801229',
 'fa_12_19791230',
 'fa_12_19801229',
 'fa_13_19791230',
```

```
'fa_13_19801229',
'fa_14_19791230',
'fa_14_19801229',
'fa_15_19791230',
'fa_15_19801229',
'fa_16_19791230',
'fa_16_19801229',
'fa_17_19791230',
'fa_17_19801229',
'fa_18_19791230',
'fa_18_19801229',
'fa_19_19791230',
'fa_19_19801229',
'fa_1_19791230',
'fa_1_19801229',
'fa_20_19791230',
'fa_20_19801229',
'fa_21_19791230',
'fa_21_19801229',
'fa_22_19791230',
'fa_22_19801229',
'fa_23_19791230',
'fa_23_19801229',
'fa_24_19791230',
'fa_24_19801229',
'fa_25_19791230',
'fa_25_19801229',
'fa_26_19791230',
'fa_26_19801229',
'fa_27_19791230',
'fa_27_19801229',
'fa_28_19791230',
'fa_28_19801229',
'fa_29_19791230',
'fa_29_19801229',
'fa_2_19791230',
'fa_2_19801229',
'fa_30_19791230',
'fa_30_19801229',
'fa_31_19791230',
'fa_31_19801229',
'fa_32_19791230',
'fa_32_19801229',
'fa_33_19791230',
'fa_33_19801229',
'fa_34_19791230',
'fa_34_19801229',
'fa_35_19791230',
```

```
'fa_35_19801229',
'fa_36_19791230',
'fa_36_19801229',
'fa_37_19791230',
'fa_37_19801229',
'fa_38_19791230',
'fa_38_19801229',
'fa_39_19791230',
'fa_39_19801229',
'fa_3_19791230',
'fa_3_19801229',
'fa_4_19791230',
'fa_4_19801229',
'fa_5_19791230',
'fa_5_19801229',
'fa_6_19791230',
'fa_6_19801229',
'fa_7_19791230',
'fa_7_19801229',
'fa_8_19791230',
'fa_8_19801229',
'fa_9_19791230',
'fa_9_19801229',
'flx_constan_19791230',
'flx_constan_19801229',
'flx_drains_19791230',
'flx_drains_19801229',
'flx_in-out_19791230',
'flx_in-out_19801229',
'flx_percent_19791230',
'flx_percent_19801229',
'flx_recharg_19791230',
'flx_recharg_19801229',
'flx_storage_19791230',
'flx_storage_19801229',
'flx_stream__19791230',
'flx_stream__19801229',
'flx_total_19791230',
'flx_total_19801229',
'flx_wells_19791230',
'flx_wells_19801229',
'fo_0_19791230',
'fo_0_19801229',
'fo_10_19791230',
'fo_10_19801229',
'fo_11_19791230',
'fo_11_19801229',
'fo_12_19791230',
```

```
'fo_12_19801229',
'fo_13_19791230',
'fo_13_19801229',
'fo_14_19791230',
'fo_14_19801229',
'fo_15_19791230',
'fo_15_19801229',
'fo_16_19791230',
'fo_16_19801229',
'fo_17_19791230',
'fo_17_19801229',
'fo_18_19791230',
'fo_18_19801229',
'fo_19_19791230',
'fo_19_19801229',
'fo_1_19791230',
'fo_1_19801229',
'fo_20_19791230',
'fo_20_19801229',
'fo_21_19791230',
'fo_21_19801229',
'fo_22_19791230',
'fo_22_19801229',
'fo_23_19791230',
'fo_23_19801229',
'fo_24_19791230',
'fo_24_19801229',
'fo_25_19791230',
'fo_25_19801229',
'fo_26_19791230',
'fo_26_19801229',
'fo_27_19791230',
'fo_27_19801229',
'fo_28_19791230',
'fo_28_19801229',
'fo_29_19791230',
'fo_29_19801229',
'fo_2_19791230',
'fo_2_19801229',
'fo_30_19791230',
'fo_30_19801229',
'fo_31_19791230',
'fo_31_19801229',
'fo_32_19791230',
'fo_32_19801229',
'fo_33_19791230',
'fo_33_19801229',
'fo_34_19791230',
```

```
'fo_34_19801229',
'fo_35_19791230',
'fo_35_19801229',
'fo_36_19791230',
'fo_36_19801229',
'fo_37_19791230',
'fo_37_19801229',
'fo_38_19791230',
'fo_38_19801229',
'fo_39_19801229',
'fo_3_19791230',
'fo_3_19801229',
'fo_4_19791230',
'fo_4_19801229',
'fo_5_19791230',
'fo_5_19801229',
'fo_6_19791230',
'fo_6_19801229',
'fo_7_19791230',
'fo_7_19801229',
'fo_8_19791230',
'fo_8_19801229',
'fo_9_19791230',
'fo_9_19801229',
'fo_hw_19791230',
'fo_hw_19801229',
'fo_tw_19791230',
'fo_tw_19801229',
'hds_00_000_000_000',
'hds_00_000_000_001',
'hds_00_000_001_000',
'hds_00_000_001_001',
'hds_00_000_002_000',
'hds_00_000_002_001',
'hds_00_000_003_000',
'hds_00_000_003_001',
'hds_00_000_004_000',
'hds_00_000_004_001',
'hds_00_000_005_000',
'hds_00_000_005_001',
'hds_00_000_006_000',
'hds_00_000_006_001',
'hds_00_000_007_000',
'hds_00_000_007_001',
'hds_00_000_008_000',
'hds_00_000_008_001',
'hds_00_000_009_000',
'hds_00_000_009_001',
```

```
'hds_00_000_010_000',
'hds_00_000_010_001',
'hds_00_000_011_000',
'hds_00_000_011_001',
'hds_00_000_012_000',
'hds_00_000_012_001',
'hds_00_000_013_000',
'hds_00_000_013_001',
'hds_00_000_014_000',
'hds_00_000_014_001',
'hds_00_000_015_000',
'hds_00_000_015_001',
'hds_00_000_016_000',
'hds_00_000_016_001',
'hds_00_000_017_000',
'hds_00_000_017_001',
'hds_00_000_018_000',
'hds_00_000_018_001',
'hds_00_000_019_000',
'hds_00_000_019_001',
'hds_00_001_000_000',
'hds_00_001_000_001',
'hds_00_001_001_000',
'hds_00_001_001_001',
'hds_00_001_002_000',
'hds_00_001_002_001',
'hds_00_001_003_000',
'hds_00_001_003_001',
'hds_00_001_004_000',
'hds_00_001_004_001',
'hds_00_001_005_000',
'hds_00_001_005_001',
'hds_00_001_006_000',
'hds_00_001_006_001',
'hds_00_001_007_000',
'hds_00_001_007_001',
'hds_00_001_008_000',
'hds_00_001_008_001',
'hds_00_001_009_000',
'hds_00_001_009_001',
'hds_00_001_010_000',
'hds_00_001_010_001',
'hds_00_001_011_000',
'hds_00_001_011_001',
'hds_00_001_012_000',
'hds_00_001_012_001',
'hds_00_001_013_000',
'hds_00_001_013_001',
```

```
'hds_00_001_014_000',
'hds_00_001_014_001',
'hds_00_001_015_000',
'hds_00_001_015_001',
'hds_00_001_016_000',
'hds_00_001_016_001',
'hds_00_001_017_000',
'hds_00_001_017_001',
'hds_00_001_018_000',
'hds_00_001_018_001',
'hds_00_001_019_000',
'hds_00_001_019_001',
'hds_00_002_000_000',
'hds_00_002_000_001',
'hds_00_002_001_000',
'hds_00_002_001_001',
'hds_00_002_002_000',
'hds_00_002_002_001',
'hds_00_002_003_000',
'hds_00_002_003_001',
'hds_00_002_004_000',
'hds_00_002_004_001',
'hds_00_002_005_000',
'hds_00_002_005_001',
'hds_00_002_006_000',
'hds_00_002_006_001',
'hds_00_002_007_000',
'hds_00_002_007_001',
'hds_00_002_008_000',
'hds_00_002_008_001',
'hds_00_002_009_001',
'hds_00_002_010_000',
'hds_00_002_010_001',
'hds_00_002_011_000',
'hds_00_002_011_001',
'hds_00_002_012_000',
'hds_00_002_012_001',
'hds_00_002_013_000',
'hds_00_002_013_001',
'hds_00_002_014_000',
'hds_00_002_014_001',
'hds_00_002_015_001',
'hds_00_002_016_000',
'hds_00_002_016_001',
'hds_00_002_017_000',
'hds_00_002_017_001',
'hds_00_002_018_000',
'hds_00_002_018_001',
```

```
'hds_00_002_019_000',
'hds_00_002_019_001',
'hds_00_003_000_000',
'hds_00_003_000_001',
'hds_00_003_001_000',
'hds_00_003_001_001',
'hds_00_003_002_000',
'hds_00_003_002_001',
'hds_00_003_003_000',
'hds_00_003_003_001',
'hds_00_003_004_000',
'hds_00_003_004_001',
'hds_00_003_005_000',
'hds_00_003_005_001',
'hds_00_003_006_000',
'hds_00_003_006_001',
'hds_00_003_007_000',
'hds_00_003_007_001',
'hds_00_003_008_001',
'hds_00_003_009_000',
'hds_00_003_009_001',
'hds_00_003_010_000',
'hds_00_003_010_001',
'hds_00_003_011_000',
'hds_00_003_011_001',
'hds_00_003_012_000',
'hds_00_003_012_001',
'hds_00_003_013_000',
'hds_00_003_013_001',
'hds_00_003_014_000',
'hds_00_003_014_001',
'hds_00_003_015_000',
'hds_00_003_015_001',
'hds_00_003_016_000',
'hds_00_003_016_001',
'hds_00_003_017_000',
'hds_00_003_017_001',
'hds_00_003_018_000',
'hds_00_003_018_001',
'hds_00_003_019_000',
'hds_00_003_019_001',
'hds_00_004_000_000',
'hds_00_004_000_001',
'hds_00_004_001_000',
'hds_00_004_001_001',
'hds_00_004_002_000',
'hds_00_004_002_001',
'hds_00_004_003_000',
```

```
'hds_00_004_003_001',
'hds_00_004_004_000',
'hds_00_004_004_001',
'hds_00_004_005_000',
'hds_00_004_005_001',
'hds_00_004_006_000',
'hds_00_004_006_001',
'hds_00_004_007_000',
'hds_00_004_007_001',
'hds_00_004_008_000',
'hds_00_004_008_001',
'hds_00_004_009_000',
'hds_00_004_009_001',
'hds_00_004_010_000',
'hds_00_004_010_001',
'hds_00_004_011_000',
'hds_00_004_011_001',
'hds_00_004_012_000',
'hds_00_004_012_001',
'hds_00_004_013_000',
'hds_00_004_013_001',
'hds_00_004_014_000',
'hds_00_004_014_001',
'hds_00_004_015_000',
'hds_00_004_015_001',
'hds_00_004_016_000',
'hds_00_004_016_001',
'hds_00_004_017_000',
'hds_00_004_017_001',
'hds_00_004_018_000',
'hds_00_004_018_001',
'hds_00_004_019_000',
'hds_00_004_019_001',
'hds_00_005_000_000',
'hds_00_005_000_001',
'hds_00_005_001_000',
'hds_00_005_001_001',
'hds_00_005_002_000',
'hds_00_005_002_001',
'hds_00_005_003_000',
'hds_00_005_003_001',
'hds_00_005_004_000',
'hds_00_005_004_001',
'hds_00_005_005_000',
'hds_00_005_005_001',
'hds_00_005_006_000',
'hds_00_005_006_001',
'hds_00_005_007_000',
```

```
'hds_00_005_007_001',
'hds_00_005_008_000',
'hds_00_005_008_001',
'hds_00_005_009_000',
'hds_00_005_009_001',
'hds_00_005_010_000',
'hds_00_005_010_001',
'hds_00_005_011_000',
'hds_00_005_011_001',
'hds_00_005_012_000',
'hds_00_005_012_001',
'hds_00_005_013_000',
'hds_00_005_013_001',
'hds_00_005_014_000',
'hds_00_005_014_001',
'hds_00_005_015_000',
'hds_00_005_015_001',
'hds_00_005_016_000',
'hds_00_005_016_001',
'hds_00_005_017_000',
'hds_00_005_017_001',
'hds_00_005_018_000',
'hds_00_005_018_001',
'hds_00_005_019_000',
'hds_00_005_019_001',
'hds_00_006_000_000',
'hds_00_006_000_001',
'hds_00_006_001_000',
'hds_00_006_001_001',
'hds_00_006_002_000',
'hds_00_006_002_001',
'hds_00_006_003_000',
'hds_00_006_003_001',
'hds_00_006_004_000',
'hds_00_006_004_001',
'hds_00_006_005_000',
'hds_00_006_005_001',
'hds_00_006_006_000',
'hds_00_006_006_001',
'hds_00_006_007_000',
'hds_00_006_007_001',
'hds_00_006_008_000',
'hds_00_006_008_001',
'hds_00_006_009_000',
'hds_00_006_009_001',
'hds_00_006_010_000',
'hds_00_006_010_001',
'hds_00_006_011_000',
```

```
'hds_00_006_011_001',
'hds_00_006_012_000',
'hds_00_006_012_001',
'hds_00_006_013_000',
'hds_00_006_013_001',
'hds_00_006_014_000',
'hds_00_006_014_001',
'hds_00_006_015_000',
'hds_00_006_015_001',
'hds_00_006_016_000',
'hds_00_006_016_001',
'hds_00_006_017_000',
'hds_00_006_017_001',
'hds_00_006_018_000',
'hds_00_006_018_001',
'hds_00_006_019_000',
'hds_00_006_019_001',
'hds_00_007_000_000',
'hds_00_007_000_001',
'hds_00_007_001_000',
'hds_00_007_001_001',
'hds_00_007_002_000',
'hds_00_007_002_001',
'hds_00_007_003_000',
'hds_00_007_003_001',
'hds_00_007_004_000',
'hds_00_007_004_001',
'hds_00_007_005_000',
'hds_00_007_005_001',
'hds_00_007_006_000',
'hds_00_007_006_001',
'hds_00_007_007_000',
'hds_00_007_007_001',
'hds_00_007_008_000',
'hds_00_007_008_001',
'hds_00_007_009_000',
'hds_00_007_009_001',
'hds_00_007_010_000',
'hds_00_007_010_001',
'hds_00_007_011_000',
'hds_00_007_011_001',
'hds_00_007_012_000',
'hds_00_007_012_001',
'hds_00_007_013_000',
'hds_00_007_013_001',
'hds_00_007_014_000',
'hds_00_007_014_001',
'hds_00_007_015_000',
```

```
'hds_00_007_015_001',
'hds_00_007_016_000',
'hds_00_007_016_001',
'hds_00_007_017_000',
'hds_00_007_017_001',
'hds_00_007_018_000',
'hds_00_007_018_001',
'hds_00_007_019_000',
'hds_00_007_019_001',
'hds_00_008_000_000',
'hds_00_008_000_001',
'hds_00_008_001_000',
'hds_00_008_001_001',
'hds_00_008_002_000',
'hds_00_008_002_001',
'hds_00_008_003_000',
'hds_00_008_003_001',
'hds_00_008_007_000',
'hds_00_008_007_001',
'hds_00_008_008_000',
'hds_00_008_008_001',
'hds_00_008_009_000',
'hds_00_008_009_001',
'hds_00_008_010_000',
'hds_00_008_010_001',
'hds_00_008_011_000',
'hds_00_008_011_001',
'hds_00_008_012_000',
'hds_00_008_012_001',
'hds_00_008_013_000',
'hds_00_008_013_001',
'hds_00_008_014_000',
'hds_00_008_014_001',
'hds_00_008_015_000',
'hds_00_008_015_001',
'hds_00_008_016_000',
'hds_00_008_016_001',
'hds_00_008_017_000',
'hds_00_008_017_001',
'hds_00_008_018_000',
'hds_00_008_018_001',
'hds_00_008_019_000',
'hds_00_008_019_001',
'hds_00_009_000_000',
'hds_00_009_000_001',
'hds_00_009_001_001',
'hds_00_009_002_000',
'hds_00_009_002_001',
```

```
'hds_00_009_003_000',
'hds_00_009_003_001',
'hds_00_009_008_000',
'hds_00_009_008_001',
'hds_00_009_009_000',
'hds_00_009_009_001',
'hds_00_009_010_000',
'hds_00_009_010_001',
'hds_00_009_011_000',
'hds_00_009_011_001',
'hds_00_009_012_000',
'hds_00_009_012_001',
'hds_00_009_013_000',
'hds_00_009_013_001',
'hds_00_009_014_000',
'hds_00_009_014_001',
'hds_00_009_015_000',
'hds_00_009_015_001',
'hds_00_009_016_000',
'hds_00_009_016_001',
'hds_00_009_017_000',
'hds_00_009_017_001',
'hds_00_009_018_000',
'hds_00_009_018_001',
'hds_00_009_019_000',
'hds_00_009_019_001',
'hds_00_010_000_000',
'hds_00_010_000_001',
'hds_00_010_001_000',
'hds_00_010_001_001',
'hds_00_010_002_000',
'hds_00_010_002_001',
'hds_00_010_003_000',
'hds_00_010_003_001',
'hds_00_010_008_000',
'hds_00_010_008_001',
'hds_00_010_009_000',
'hds_00_010_009_001',
'hds_00_010_010_000',
'hds_00_010_010_001',
'hds_00_010_011_000',
'hds_00_010_011_001',
'hds_00_010_012_000',
'hds_00_010_012_001',
'hds_00_010_013_000',
'hds_00_010_013_001',
'hds_00_010_014_000',
'hds_00_010_014_001',
```

```
'hds_00_010_015_000',
'hds_00_010_015_001',
'hds_00_010_016_000',
'hds_00_010_016_001',
'hds_00_010_017_000',
'hds_00_010_017_001',
'hds_00_010_018_000',
'hds_00_010_018_001',
'hds_00_010_019_000',
'hds_00_010_019_001',
'hds_00_011_000_000',
'hds_00_011_000_001',
'hds_00_011_001_000',
'hds_00_011_001_001',
'hds_00_011_002_000',
'hds_00_011_002_001',
'hds_00_011_003_000',
'hds_00_011_003_001',
'hds_00_011_008_000',
'hds_00_011_008_001',
'hds_00_011_009_000',
'hds_00_011_009_001',
'hds_00_011_010_000',
'hds_00_011_010_001',
'hds_00_011_011_000',
'hds_00_011_011_001',
'hds_00_011_012_000',
'hds_00_011_012_001',
'hds_00_011_013_000',
'hds_00_011_013_001',
'hds_00_011_014_000',
'hds_00_011_014_001',
'hds_00_011_015_000',
'hds_00_011_015_001',
'hds_00_011_016_000',
'hds_00_011_016_001',
'hds_00_011_017_000',
'hds_00_011_017_001',
'hds_00_011_018_000',
'hds_00_011_018_001',
'hds_00_011_019_000',
'hds_00_011_019_001',
'hds_00_012_000_000',
'hds_00_012_000_001',
'hds_00_012_001_000',
'hds_00_012_001_001',
'hds_00_012_002_000',
'hds_00_012_002_001',
```

```
'hds_00_012_003_000',
'hds_00_012_003_001',
'hds_00_012_008_000',
'hds_00_012_008_001',
'hds_00_012_009_000',
'hds_00_012_009_001',
'hds_00_012_010_000',
'hds_00_012_010_001',
'hds_00_012_011_000',
'hds_00_012_011_001',
'hds_00_012_012_000',
'hds_00_012_012_001',
'hds_00_012_013_000',
'hds_00_012_013_001',
'hds_00_012_014_000',
'hds_00_012_014_001',
'hds_00_012_015_000',
'hds_00_012_015_001',
'hds_00_012_016_000',
'hds_00_012_016_001',
'hds_00_012_017_000',
'hds_00_012_017_001',
'hds_00_012_018_000',
'hds_00_012_018_001',
'hds_00_012_019_000',
'hds_00_012_019_001',
'hds_00_013_000_000',
'hds_00_013_000_001',
'hds_00_013_001_000',
'hds_00_013_001_001',
'hds_00_013_003_000',
'hds_00_013_003_001',
'hds_00_013_008_000',
'hds_00_013_008_001',
'hds_00_013_009_000',
'hds_00_013_009_001',
'hds_00_013_010_001',
'hds_00_013_011_000',
'hds_00_013_011_001',
'hds_00_013_012_000',
'hds_00_013_012_001',
'hds_00_013_013_000',
'hds_00_013_013_001',
'hds_00_013_014_000',
'hds_00_013_014_001',
'hds_00_013_015_000',
'hds_00_013_015_001',
'hds_00_013_016_000',
```

```
'hds_00_013_016_001',
'hds_00_013_017_000',
'hds_00_013_017_001',
'hds_00_013_018_000',
'hds_00_013_018_001',
'hds_00_013_019_000',
'hds_00_013_019_001',
'hds_00_014_000_000',
'hds_00_014_000_001',
'hds_00_014_001_000',
'hds_00_014_001_001',
'hds_00_014_002_000',
'hds_00_014_002_001',
'hds_00_014_003_000',
'hds_00_014_003_001',
'hds_00_014_008_000',
'hds_00_014_008_001',
'hds_00_014_009_000',
'hds_00_014_009_001',
'hds_00_014_010_000',
'hds_00_014_010_001',
'hds_00_014_011_000',
'hds_00_014_011_001',
'hds_00_014_012_000',
'hds_00_014_012_001',
'hds_00_014_013_000',
'hds_00_014_013_001',
'hds_00_014_014_000',
'hds_00_014_014_001',
'hds_00_014_015_000',
'hds_00_014_015_001',
'hds_00_014_016_000',
'hds_00_014_016_001',
'hds_00_014_017_000',
'hds_00_014_017_001',
'hds_00_014_018_000',
'hds_00_014_018_001',
'hds_00_014_019_000',
'hds_00_014_019_001',
'hds_00_015_000_000',
'hds_00_015_000_001',
'hds_00_015_001_000',
'hds_00_015_001_001',
'hds_00_015_002_000',
'hds_00_015_002_001',
'hds_00_015_003_000',
'hds_00_015_003_001',
'hds_00_015_008_000',
```

```
'hds_00_015_008_001',
'hds_00_015_009_000',
'hds_00_015_009_001',
'hds_00_015_010_000',
'hds_00_015_010_001',
'hds_00_015_011_000',
'hds_00_015_011_001',
'hds_00_015_012_000',
'hds_00_015_012_001',
'hds_00_015_013_000',
'hds_00_015_013_001',
'hds_00_015_014_000',
'hds_00_015_014_001',
'hds_00_015_015_000',
'hds_00_015_015_001',
'hds_00_015_016_001',
'hds_00_015_017_000',
'hds_00_015_017_001',
'hds_00_015_018_000',
'hds_00_015_018_001',
'hds_00_015_019_000',
'hds_00_015_019_001',
'hds_00_016_000_000',
'hds_00_016_000_001',
'hds_00_016_001_000',
'hds_00_016_001_001',
'hds_00_016_002_000',
'hds_00_016_002_001',
'hds_00_016_003_000',
'hds_00_016_003_001',
'hds_00_016_008_000',
'hds_00_016_008_001',
'hds_00_016_009_000',
'hds_00_016_009_001',
'hds_00_016_010_000',
'hds_00_016_010_001',
'hds_00_016_011_000',
'hds_00_016_011_001',
'hds_00_016_012_000',
'hds_00_016_012_001',
'hds_00_016_013_000',
'hds_00_016_013_001',
'hds_00_016_014_000',
'hds_00_016_014_001',
'hds_00_016_015_000',
'hds_00_016_015_001',
'hds_00_016_016_000',
'hds_00_016_016_001',
```

```
'hds_00_016_017_000',
'hds_00_016_017_001',
'hds_00_016_018_000',
'hds_00_016_018_001',
'hds_00_016_019_000',
'hds_00_016_019_001',
'hds_00_017_000_000',
'hds_00_017_000_001',
'hds_00_017_001_000',
'hds_00_017_001_001',
'hds_00_017_002_000',
'hds_00_017_002_001',
'hds_00_017_003_000',
'hds_00_017_003_001',
'hds_00_017_008_000',
'hds_00_017_008_001',
'hds_00_017_009_000',
'hds_00_017_009_001',
'hds_00_017_010_000',
'hds_00_017_010_001',
'hds_00_017_011_000',
'hds_00_017_011_001',
'hds_00_017_012_000',
'hds_00_017_012_001',
'hds_00_017_013_000',
'hds_00_017_013_001',
'hds_00_017_014_000',
'hds_00_017_014_001',
'hds_00_017_015_000',
'hds_00_017_015_001',
'hds_00_017_016_000',
'hds_00_017_016_001',
'hds_00_017_017_000',
'hds_00_017_017_001',
'hds_00_017_018_000',
'hds_00_017_018_001',
'hds_00_017_019_000',
'hds_00_017_019_001',
'hds_00_018_000_000',
'hds_00_018_000_001',
'hds_00_018_001_000',
'hds_00_018_001_001',
'hds_00_018_002_000',
'hds_00_018_002_001',
'hds_00_018_003_000',
'hds_00_018_003_001',
'hds_00_018_008_000',
'hds_00_018_008_001',
```

```
'hds_00_018_009_000',
'hds_00_018_009_001',
'hds_00_018_010_000',
'hds_00_018_010_001',
'hds_00_018_011_000',
'hds_00_018_011_001',
'hds_00_018_012_000',
'hds_00_018_012_001',
'hds_00_018_013_000',
'hds_00_018_013_001',
'hds_00_018_014_000',
'hds_00_018_014_001',
'hds_00_018_015_000',
'hds_00_018_015_001',
'hds_00_018_016_000',
'hds_00_018_016_001',
'hds_00_018_017_000',
'hds_00_018_017_001',
'hds_00_018_018_000',
'hds_00_018_018_001',
'hds_00_018_019_000',
'hds_00_018_019_001',
'hds_00_019_000_000',
'hds_00_019_000_001',
'hds_00_019_001_000',
'hds_00_019_001_001',
'hds_00_019_002_000',
'hds_00_019_002_001',
'hds_00_019_003_000',
'hds_00_019_003_001',
'hds_00_019_008_000',
'hds_00_019_008_001',
'hds_00_019_009_000',
'hds_00_019_009_001',
'hds_00_019_010_000',
'hds_00_019_010_001',
'hds_00_019_011_000',
'hds_00_019_011_001',
'hds_00_019_012_000',
'hds_00_019_012_001',
'hds_00_019_013_000',
'hds_00_019_013_001',
'hds_00_019_014_000',
'hds_00_019_014_001',
'hds_00_019_015_000',
'hds_00_019_015_001',
'hds_00_019_016_000',
'hds_00_019_016_001',
```

```
'hds_00_019_017_000',
'hds_00_019_017_001',
'hds_00_019_018_000',
'hds_00_019_018_001',
'hds_00_019_019_000',
'hds_00_019_019_001',
'hds_00_020_000_000',
'hds_00_020_000_001',
'hds_00_020_001_000',
'hds_00_020_001_001',
'hds_00_020_002_000',
'hds_00_020_002_001',
'hds_00_020_003_000',
'hds_00_020_003_001',
'hds_00_020_008_000',
'hds_00_020_008_001',
'hds_00_020_009_000',
'hds_00_020_009_001',
'hds_00_020_010_000',
'hds_00_020_010_001',
'hds_00_020_011_000',
'hds_00_020_011_001',
'hds_00_020_012_000',
'hds_00_020_012_001',
'hds_00_020_013_000',
'hds_00_020_013_001',
'hds_00_020_014_000',
'hds_00_020_014_001',
'hds_00_020_015_000',
'hds_00_020_015_001',
'hds_00_020_016_000',
'hds_00_020_016_001',
'hds_00_020_017_000',
'hds_00_020_017_001',
'hds_00_020_018_000',
'hds_00_020_018_001',
'hds_00_020_019_000',
'hds_00_020_019_001',
'hds_00_021_000_000',
'hds_00_021_000_001',
'hds_00_021_001_000',
'hds_00_021_001_001',
'hds_00_021_002_000',
'hds_00_021_002_001',
'hds_00_021_003_000',
'hds_00_021_003_001',
'hds_00_021_006_000',
'hds_00_021_006_001',
```

```
'hds_00_021_007_000',
'hds_00_021_007_001',
'hds_00_021_008_000',
'hds_00_021_008_001',
'hds_00_021_009_000',
'hds_00_021_009_001',
'hds_00_021_010_001',
'hds_00_021_011_000',
'hds_00_021_011_001',
'hds_00_021_012_000',
'hds_00_021_012_001',
'hds_00_021_013_000',
'hds_00_021_013_001',
'hds_00_021_014_000',
'hds_00_021_014_001',
'hds_00_021_015_000',
'hds_00_021_015_001',
'hds_00_021_016_000',
'hds_00_021_016_001',
'hds_00_021_017_000',
'hds_00_021_017_001',
'hds_00_021_018_000',
'hds_00_021_018_001',
'hds_00_021_019_000',
'hds_00_021_019_001',
'hds_00_022_000_000',
'hds_00_022_000_001',
'hds_00_022_001_000',
'hds_00_022_001_001',
'hds_00_022_002_000',
'hds_00_022_002_001',
'hds_00_022_003_000',
'hds_00_022_003_001',
'hds_00_022_004_000',
'hds_00_022_004_001',
'hds_00_022_006_000',
'hds_00_022_006_001',
'hds_00_022_007_000',
'hds_00_022_007_001',
'hds_00_022_008_000',
'hds_00_022_008_001',
'hds_00_022_009_000',
'hds_00_022_009_001',
'hds_00_022_010_000',
'hds_00_022_010_001',
'hds_00_022_011_000',
'hds_00_022_011_001',
'hds_00_022_012_000',
```

```
 'hds_00_022_012_001',
 'hds_00_022_013_000',
 'hds_00_022_013_001',
 'hds_00_022_014_000',
 'hds_00_022_014_001',
 'hds_00_022_015_001',
 'hds_00_022_016_000',
 'hds_00_022_016_001',
 'hds_00_022_017_000',
 'hds_00_022_017_001',
 'hds_00_022_018_000',
 'hds_00_022_018_001',
 'hds_00_022_019_000',
 'hds_00_022_019_001',
 'hds_00_023_000_000',
 'hds_00_023_000_001',
 'hds_00_023_001_000',
 'hds_00_023_001_001',
 'hds_00_023_002_000',
 'hds_00_023_002_001',
 'hds_00_023_003_000',
 'hds_00_023_003_001',
 'hds_00_023_004_000',
 'hds_00_023_004_001',
 'hds_00_023_005_000',
 'hds_00_023_005_001',
 'hds_00_023_006_000',
 'hds_00_023_006_001',
 'hds_00_023_007_000',
 'hds_00_023_007_001',
 'hds_00_023_008_000',
 ...]
```

We can therefore repeat above analysis for the observations that currently have zero weight by turning those observations "on".

Beware: calculating the Schur complement for all potential observation types and locations could take some time!! So we will sample to speed things up. You may need to further reduce the number of potential obs - you can do this by adding [0::2] to take every second element for example.

```
In [27]: new_obs = [x for x in z_obs if "hds_00" in x]#and x.endswith("_000")  # all heads in
         print("number of new potential head observation locations considered: {}".format(len(n
```

```
number of new potential head observation locations considered: 1395
```

```
In [28]: from datetime import datetime
         start = datetime.now()
```

```
df_worth_new = sc.get_added_obs_importance(obslist_dict=new_obs, base_obslist=sc.pst.r
print("took:",datetime.now() - start)
```

took: 0:02:46.098102

In [29]:
```
df_worth_new.head()
```

Out[29]:

```
                      fa_hw_19791230  fa_hw_19801229  fa_tw_19791230  \
base                    48708.787021   277258.059336    22609.793588
hds_00_000_000_000      48680.325274   277236.472932    22609.717448
hds_00_000_000_001      48676.577443   275221.256112    22608.768505
hds_00_000_001_000      48681.561424   277236.427929    22609.645559
hds_00_000_001_001      48678.434575   275165.006823    22608.968020

                      fa_tw_19801229  hds_00_013_002_000  hds_00_013_002_001  \
base                   220463.794436            0.111608            0.192769
hds_00_000_000_000     220463.630209            0.111602            0.192745
hds_00_000_000_001     218435.629344            0.111608            0.191419
hds_00_000_001_000     220463.643286            0.111598            0.192738
hds_00_000_001_001     218391.377944            0.111607            0.191471

                      part_status     part_time
base                          0.0  93226.847888
hds_00_000_000_000            0.0  91552.821262
hds_00_000_000_001            0.0  91946.009838
hds_00_000_001_000            0.0  91612.117694
hds_00_000_001_001            0.0  91987.119582
```

1.1.5 nice! now let's process a little bit and make some plots of (potential) data worth

In [30]:
```
def worth_plot_prep(df):
    # some processing
    df_new_base = df.loc["base",:].copy()  # "base" row
    df_new_imax = df.apply(lambda x: df_new_base - x, axis=1).idxmax()  # obs with la
    df_new_worth = 100.0 * (df.apply(lambda x: df_new_base - x, axis=1) / df_new_base)

    # plot prep
    df_new_worth_plot = df_new_worth[df_new_worth.index != 'base'].copy()
    df_new_worth_plot.loc[:,'names'] = df_new_worth_plot.index
    names = df_new_worth_plot.names
    df_new_worth_plot.loc[:,"i"] = names.apply(lambda x: int(x[8:10]))
    df_new_worth_plot.loc[:,"j"] = names.apply(lambda x: int(x[11:14]))
    df_new_worth_plot.loc[:,'kper'] = names.apply(lambda x: int(x[-3:]))
    #df_new_worth_plot.head()

    return df_new_worth_plot, df_new_imax
```

In [31]:
```
df_worth_new_plot, df_worth_new_imax = worth_plot_prep(df_worth_new)
```

In [32]:
```
df_worth_new_plot.head()
```

Out[32]:
```
                    fa_hw_19791230  fa_hw_19801229  fa_tw_19791230  \
hds_00_000_000_000        0.058432        0.007786        0.000337
hds_00_000_000_001        0.066127        0.734624        0.004534
hds_00_000_001_000        0.055895        0.007802        0.000655
hds_00_000_001_001        0.062314        0.754911        0.003651
hds_00_000_002_000        0.055161        0.007937        0.000654

                    fa_tw_19801229  hds_00_013_002_000  hds_00_013_002_001  \
hds_00_000_000_000        0.000074            0.005094            0.012494
hds_00_000_000_001        0.919954            0.000115            0.700256
hds_00_000_001_000        0.000069            0.009101            0.016127
hds_00_000_001_001        0.940026            0.001312            0.673578
hds_00_000_002_000        0.000066            0.018307            0.022293

                    part_status  part_time               names  i  j  kper
hds_00_000_000_000          NaN   1.795649  hds_00_000_000_000  0  0     0
hds_00_000_000_001          NaN   1.373894  hds_00_000_000_001  0  0     1
hds_00_000_001_000          NaN   1.732044  hds_00_000_001_000  0  1     0
hds_00_000_001_001          NaN   1.329798  hds_00_000_001_001  0  1     1
hds_00_000_002_000          NaN   1.577571  hds_00_000_002_000  0  2     0
```

In [33]:
```
df_worth_new_imax  # which obs causes largest unc var reduction?
```

Out[33]:
```
fa_hw_19791230        hds_00_009_016_001
fa_hw_19801229        hds_00_011_013_001
fa_tw_19791230        hds_00_020_014_001
fa_tw_19801229        hds_00_026_010_001
hds_00_013_002_000    hds_00_016_001_001
hds_00_013_002_001    hds_00_017_002_001
part_status                         base
part_time             hds_00_017_002_001
dtype: object
```

In [34]:
```
df_worth_new_plot.drop(axis=1,labels=["part_status"],inplace=True) # drop "part_statu
df_worth_new_plot.head()
```

Out[34]:
```
                    fa_hw_19791230  fa_hw_19801229  fa_tw_19791230  \
hds_00_000_000_000        0.058432        0.007786        0.000337
hds_00_000_000_001        0.066127        0.734624        0.004534
hds_00_000_001_000        0.055895        0.007802        0.000655
hds_00_000_001_001        0.062314        0.754911        0.003651
hds_00_000_002_000        0.055161        0.007937        0.000654

                    fa_tw_19801229  hds_00_013_002_000  hds_00_013_002_001  \
hds_00_000_000_000        0.000074            0.005094            0.012494
hds_00_000_000_001        0.919954            0.000115            0.700256
hds_00_000_001_000        0.000069            0.009101            0.016127
```

```
hds_00_000_001_001        0.940026             0.001312             0.673578
hds_00_000_002_000        0.000066             0.018307             0.022293

                    part_time               names  i  j  kper
hds_00_000_000_000   1.795649  hds_00_000_000_000  0  0     0
hds_00_000_000_001   1.373894  hds_00_000_000_001  0  0     1
hds_00_000_001_000   1.732044  hds_00_000_001_000  0  1     0
hds_00_000_001_001   1.329798  hds_00_000_001_001  0  1     1
hds_00_000_002_000   1.577571  hds_00_000_002_000  0  2     0
```

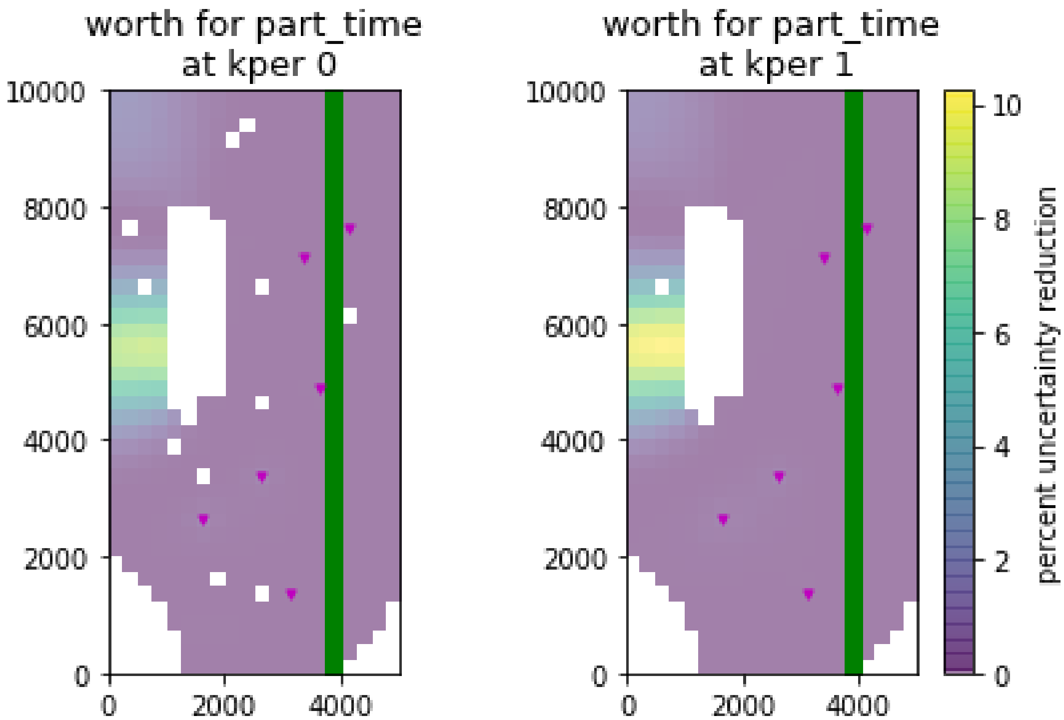
1.1.6 plotting

```
In [35]: m = flopy.modflow.Modflow.load("freyberg.nam", model_ws=os.path.join(m_d))
```

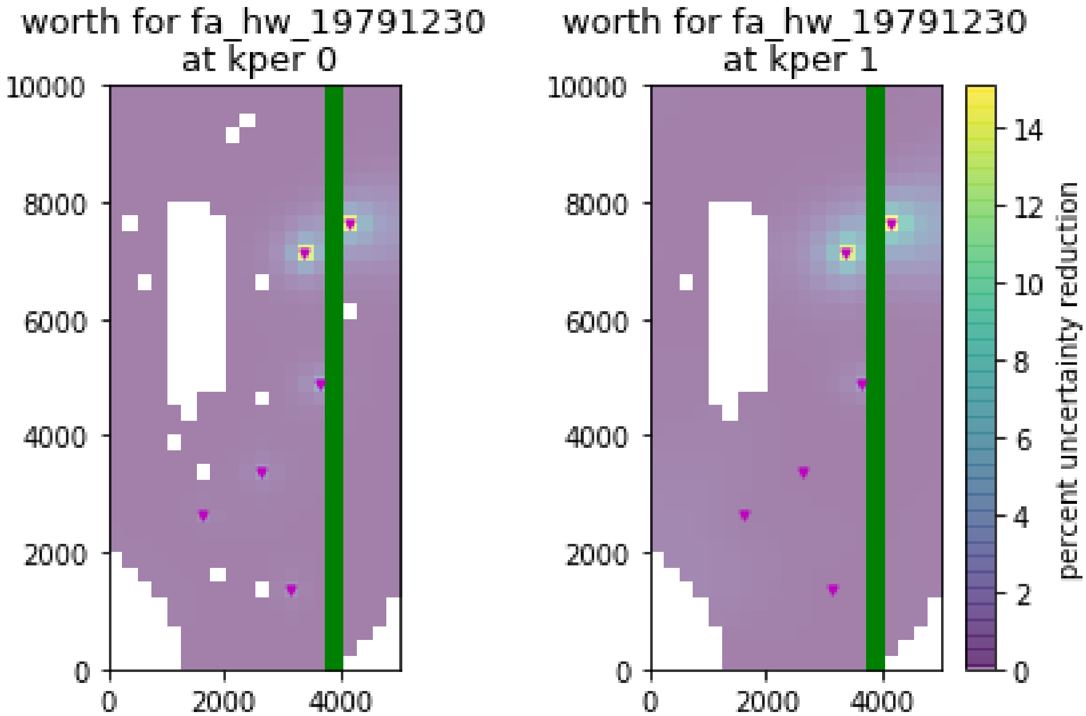
```
In [36]: def plot_added_importance(df_worth_plot, ml, forecast_name=None,
                                   newlox=None,):

             vmax = df_worth_plot[forecast_name].max()

             fig, axs = plt.subplots(1,2)
             if newlox:
                 currx = []
                 curry = []
                 for i,clox in enumerate(newlox):
                     crow = int(clox[8:10])
                     ccol = int(clox[11:14])
                     currx.append(ml.sr.xcentergrid[crow,ccol])
                     curry.append(ml.sr.ycentergrid[crow,ccol])

             for sp,ax in enumerate(axs): # by kpers
                 unc_array = np.zeros_like(ml.upw.hk[0].array) - 1
                 df_worth_csp = df_worth_plot.groupby('kper').get_group(sp)
                 for i,j,unc in zip(df_worth_csp.i,df_worth_csp.j,
                                    df_worth_csp[forecast_name]):
                     unc_array[i,j] = unc
                 unc_array[unc_array == -1] = np.NaN
                 cb = ax.imshow(unc_array,interpolation="nearest",
                                alpha=0.5,extent=ml.sr.get_extent(),
                                vmin=0, vmax=vmax)
                 if sp==1:
                     plt.colorbar(cb,label="percent uncertainty reduction")

                 # plot sfr
                 sfr_data = ml.sfr.stress_period_data[0]
                 sfr_x = ml.sr.xcentergrid[sfr_data["i"],sfr_data["j"]]
                 sfr_y = ml.sr.ycentergrid[sfr_data["i"],sfr_data["j"]]
                 for (x,y) in zip(sfr_x,sfr_y):
                     ax.scatter([x],[y],marker="s",color="g",s=26)
```

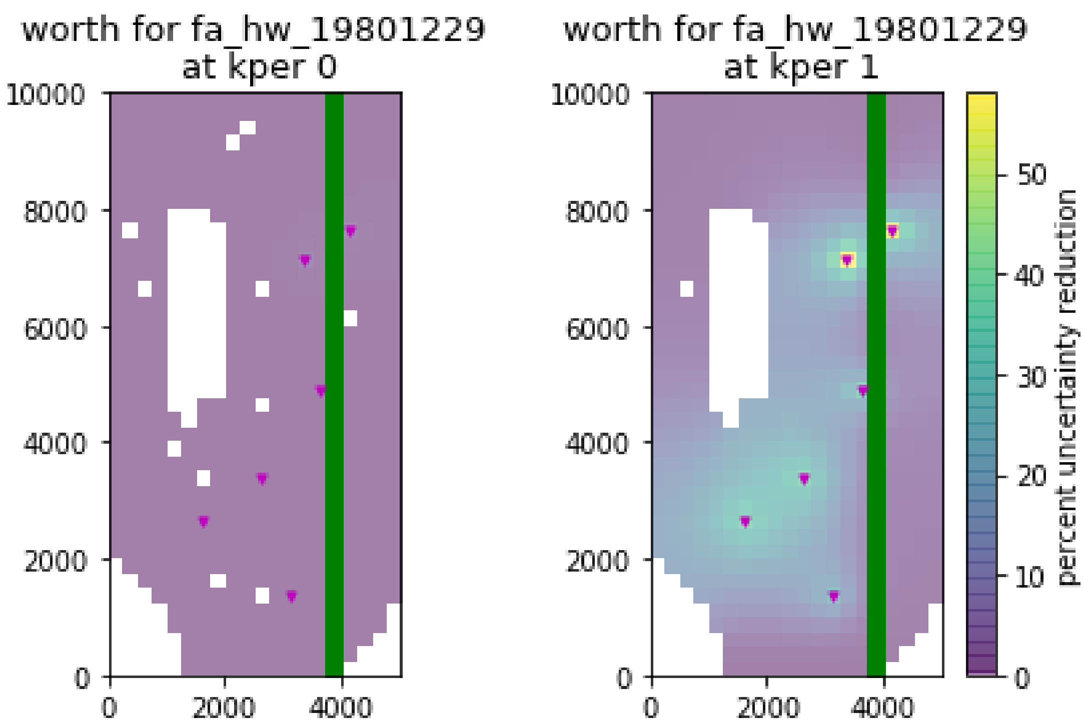
```
        # plot the pumping wells
        wel_data = ml.wel.stress_period_data[0]
        wel_x = ml.sr.xcentergrid[wel_data["i"],wel_data["j"]]
        wel_y = ml.sr.ycentergrid[wel_data["i"],wel_data["j"]]
        for w,(x,y) in enumerate(zip(wel_x,wel_y)):
            ax.scatter([x],[y],marker="v",color="m",s=10)

        if newlox:
            for nl,(cx,cy,cobs) in enumerate(zip(currx, curry, newlox)):
                csp = int(cobs[-1])
                if csp == sp:
                    ax.plot(cx, cy, 'rd', mfc=None, ms=10, alpha=0.8)
                    ax.text(cx-50,cy-50, nl, size=10)

        # plot the location of the forecast if possible
        if forecast_name.startswith('hds'):
            i = int(forecast_name[8:10])
            j = int(forecast_name[11:14])
            forecast_x = ml.sr.xcentergrid[i,j]
            forecast_y = ml.sr.ycentergrid[i,j]
            ax.scatter(forecast_x, forecast_y, marker='o', s=600, alpha=0.5)

        ax.set_title("worth for {0}\n at kper {1}".format(forecast_name,sp), fontsize=
        plt.tight_layout()
    return fig
```

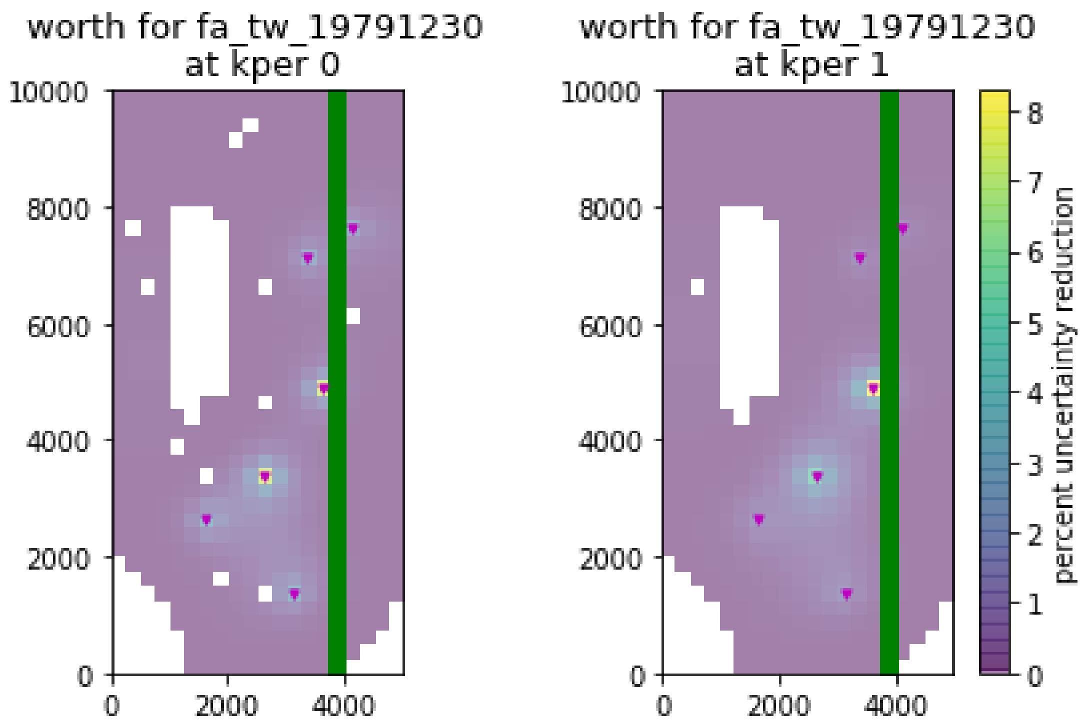
```
In [37]: fig = plot_added_importance(df_worth_plot=df_worth_new_plot, ml=m,forecast_name="part_
```

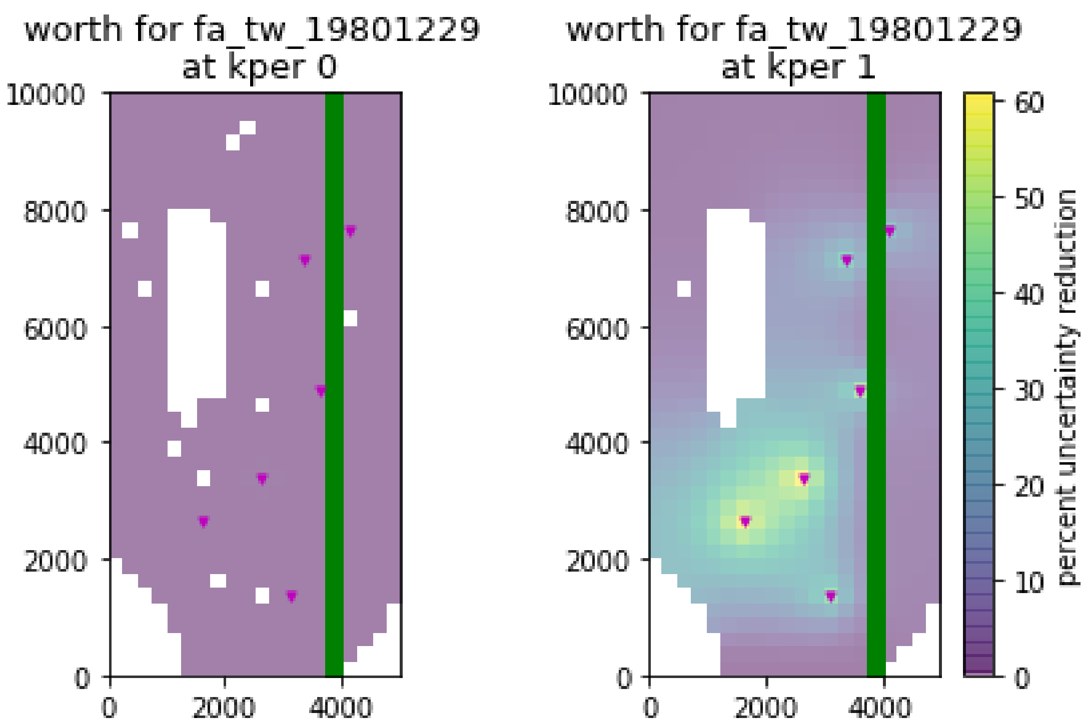


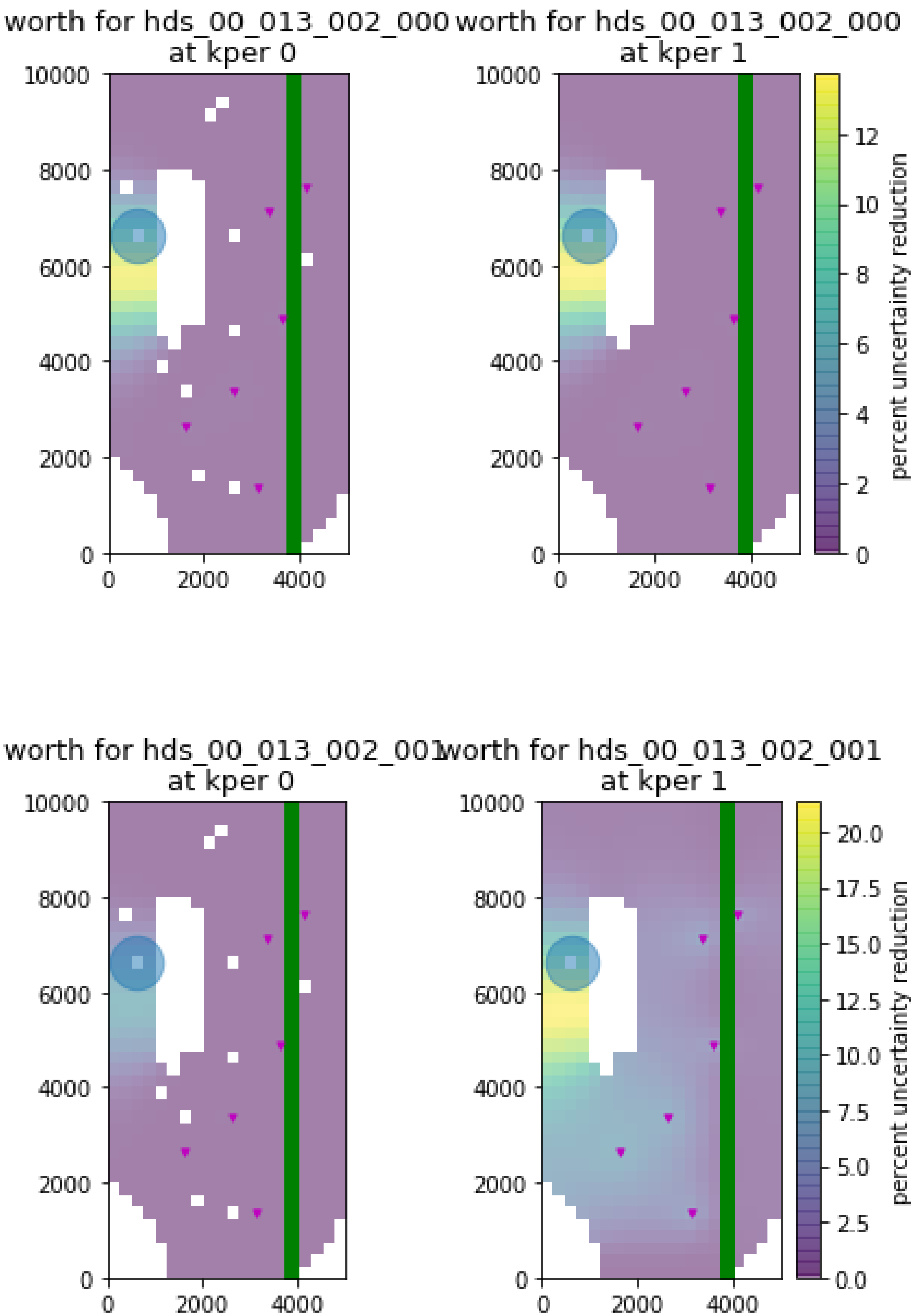

```
In [38]: for i in [x for x in forecasts if "part_status" not in x]:
             fig = plot_added_importance(df_worth_plot=df_worth_new_plot, ml=m,
                                         forecast_name=i)
             #fig.savefig('add_worth_{}.pdf'.format(i))
```

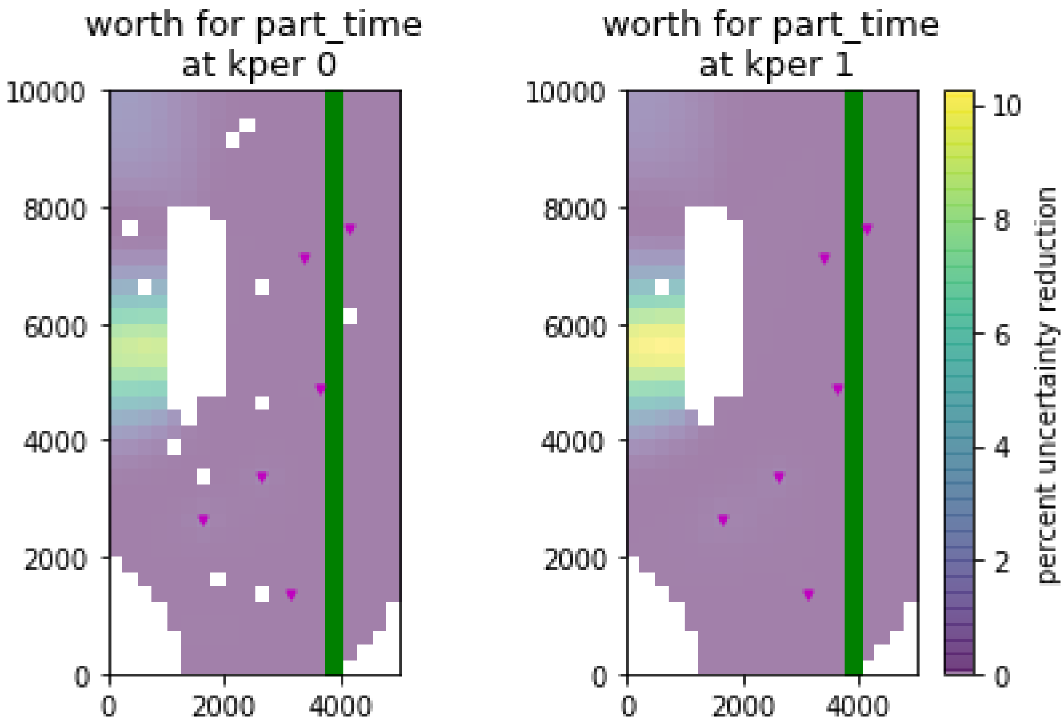

worth for fa_hw_19791230
at kper 0
worth for fa_hw_19791230
at kper 1
10000
8000
6000
4000
2000
0
0
2000
4000
14
12
10
8
6
4
2
0
percent uncertainty reduction


worth for fa_hw_19801229
at kper 0
worth for fa_hw_19801229
at kper 1
10000
8000
6000
4000
2000
0
0
2000
4000
50
40
30
20
10
0
percent uncertainty reduction


worth for fa_tw_19791230
at kper 0
worth for fa_tw_19791230
at kper 1
percent uncertainty reduction


worth for fa_tw_19801229
at kper 0
worth for fa_tw_19801229
at kper 1
percent uncertainty reduction


worth for hds_00_013_002_000
at kper 0
worth for hds_00_013_002_000
at kper 1
percent uncertainty reduction
worth for hds_00_013_002_001
at kper 0
worth for hds_00_013_002_001
at kper 1
percent uncertainty reduction



1.2 the "next best" observation

This is what we would ultimately like to know... Takes into account what we already know through incrementally making additional observations. For example, consider making an observation in the middle of the zone of highest worth. Where should we subsequently collect data?

Let's just use the same potential observation list for now (the head in every top-layer cell) and evaluate which ones to collect, if we only had the budget for 5, in the context of the particle travel time prediction.

```
In [39]: start = datetime.now()
         next_most_df = sc.next_most_important_added_obs(forecast='part_time',niter=5,obslist_c
                                                         base_obslist=sc.pst.nnz_obs_names,rese
         print("took:",datetime.now() - start)
```

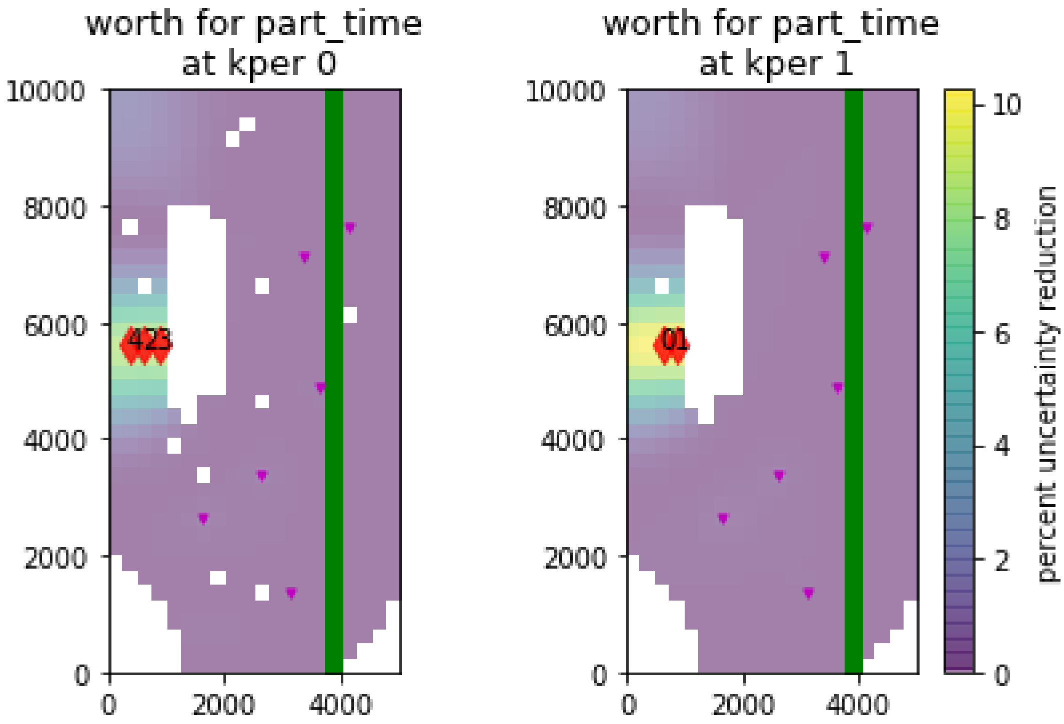
```
took: 0:11:08.660878
```

```
In [40]: next_most_df
```

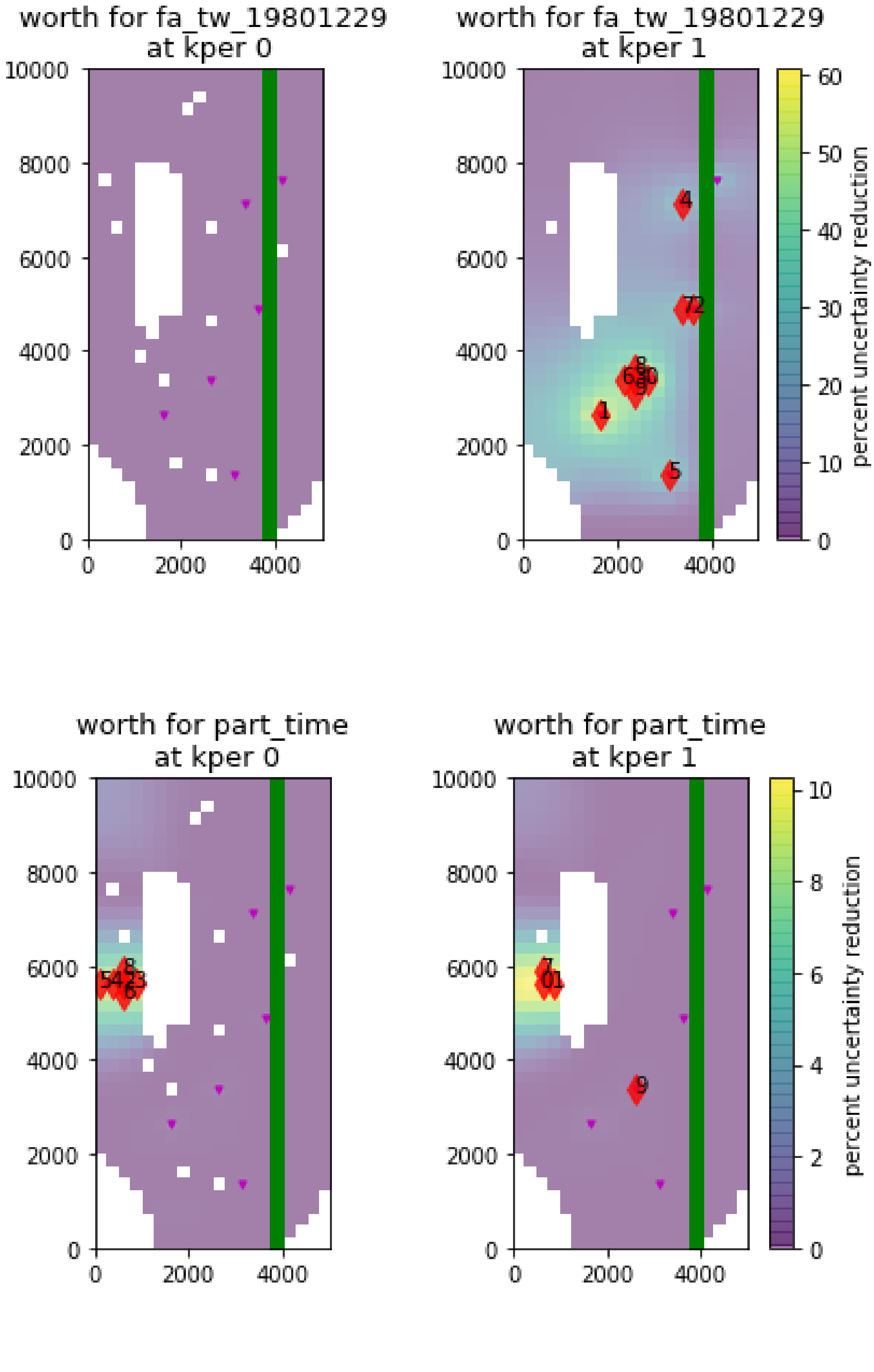
```
Out[40]:                                     best_obs  part_time_variance  \
         hds_00_017_002_001  hds_00_017_002_001        83678.784373
         hds_00_017_003_001  hds_00_017_003_001        78000.012459
         hds_00_017_002_000  hds_00_017_002_000        73651.906388
         hds_00_017_003_000  hds_00_017_003_000        70311.969583
```

```
hds_00_017_001_000  hds_00_017_001_000       67685.623078

                    unc_reduce_iter_base  unc_reduce_initial_base
hds_00_017_002_001             10.241753                10.241753
hds_00_017_003_001              6.786394                16.333101
hds_00_017_002_000              5.574494                20.997108
hds_00_017_003_000              4.534759                24.579699
hds_00_017_001_000              3.735277                27.396856
```

In [41]:
```
fig = plot_added_importance(df_worth_new_plot, m, 'part_time',
                            newlox=next_most_df.best_obs.tolist())
```



In [42]:
```
# for fun after class - this will take a while!
for i in ["fa_tw_19801229","part_time"]:#[x for x in forecasts if "part_status" not i
    next_most_df = sc.next_most_important_added_obs(forecast=i,niter=10,obslist_dict=c
                                                    base_obslist=sc.pst.nnz_obs_names
    fig = plot_added_importance(df_worth_new_plot, m, forecast_name=i,
                                newlox=next_most_df.best_obs.tolist())
    fig.savefig('next_best_10_worth_{}.pdf'.format(i))
```


worth for fa_tw_19801229
at kper 0
worth for fa_tw_19801229
at kper 1
percent uncertainty reduction
0
10
20
30
40
50
60
0
2000
4000
6000
8000
10000
worth for part_time
at kper 0
worth for part_time
at kper 1
percent uncertainty reduction
0
2
4
6
8
10

1.2.1 Note: an important assumption underpinning the above is that the model is able to fit observations to a level that is commensurate with measurement noise... Are we comfortable with this assumption? We will discuss this more in pestpp-glm_part2.ipynb

In [43]:
```
# recall...
pst.observation_data.loc[pst.nnz_obs_names,:]
```

Out[43]:

obsnme	obsnme	obsval	weight	obgnme	extra
fo_39_19791230	fo_39_19791230	11979.405235	0.01	calflux	NaN
hds_00_002_009_000	hds_00_002_009_000	37.485633	4.00	calhead	NaN
hds_00_002_015_000	hds_00_002_015_000	35.439932	4.00	calhead	NaN
hds_00_003_008_000	hds_00_003_008_000	38.297646	4.00	calhead	NaN
hds_00_009_001_000	hds_00_009_001_000	41.398610	4.00	calhead	NaN
hds_00_013_010_000	hds_00_013_010_000	35.670433	4.00	calhead	NaN
hds_00_015_016_000	hds_00_015_016_000	35.184689	4.00	calhead	NaN
hds_00_021_010_000	hds_00_021_010_000	35.792243	4.00	calhead	NaN
hds_00_022_015_000	hds_00_022_015_000	34.595529	4.00	calhead	NaN
hds_00_024_004_000	hds_00_024_004_000	37.360729	4.00	calhead	NaN
hds_00_026_006_000	hds_00_026_006_000	36.482048	4.00	calhead	NaN
hds_00_029_015_000	hds_00_029_015_000	34.903695	4.00	calhead	NaN
hds_00_033_007_000	hds_00_033_007_000	35.583158	4.00	calhead	NaN
hds_00_034_010_000	hds_00_034_010_000	34.610165	4.00	calhead	NaN

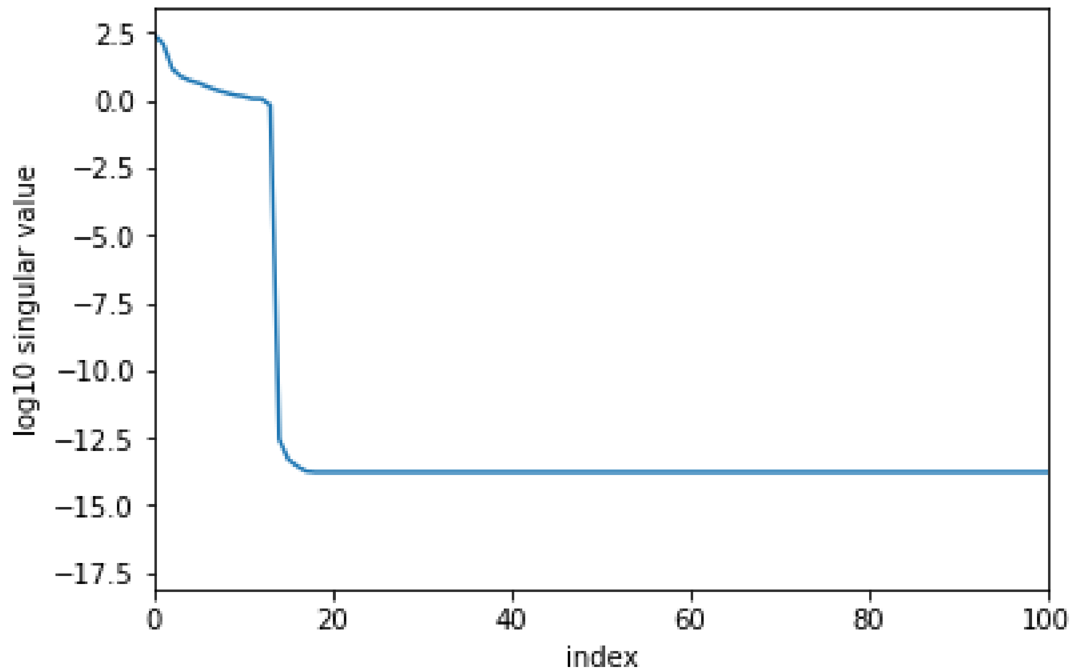
1.2.2 an "extra" if we have time: parameter identifiability

In [44]:
```
la = pyemu.ErrVar(jco=jco)
```

In [45]:
```
s = la.qhalfx.s  # singular spectrum
s.x[:10]
```

Out[45]:
```
array([[258.05273352],
       [121.52704432],
       [ 14.808986  ],
       [  7.51386804],
       [  5.34060963],
       [  4.31992965],
       [  3.13423815],
       [  2.35873769],
       [  1.90481097],
       [  1.56222005]])
```

In [46]:
```
figure = plt.figure()
ax = plt.subplot(111)
ax.plot(np.log10(s.x))
ax.set_ylabel("log10 singular value")
ax.set_xlabel("index")
ax.set_xlim(0,100)
plt.show()
```



As expected, singluar spectrum decays rapidly.

```
In [47]: truncation_thresh = 1e-6
         n_signif_singvals = ((s.x / s[0].x) > 1e-6).sum()
         n_signif_singvals
```

```
Out[47]: 14
```

```
In [48]: print("This means that, on the basis of the {0} (non-zero) weighted observations, \
         there are {1} unique pieces of information in the calibration dataset.  \
         Recall the inverse problem we are trying to solve involves the estimation of {2} param
               format(pst.nnz_obs, n_signif_singvals, pst.npar_adj))
```

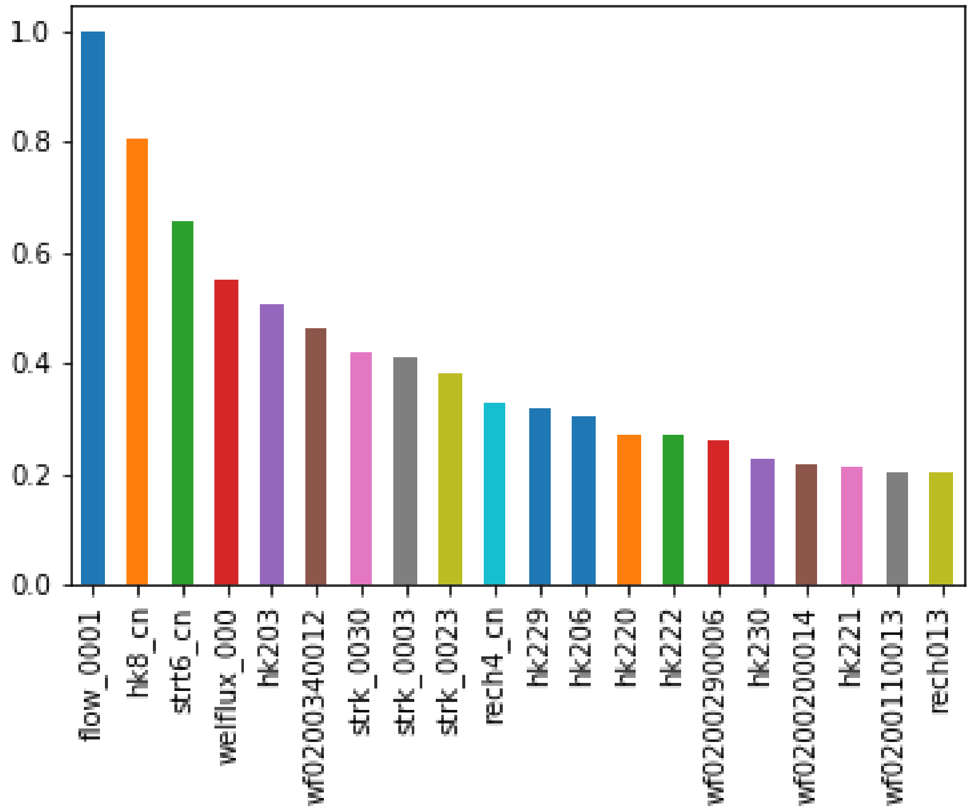
```
This means that, on the basis of the 14 (non-zero) weighted observations, there are 14 unique p
```

Now let's compute the identifiability of actual model parameters based on these singular vectors. Identifiability ranges from 0 (not identified by the data) to 1 (full identified by the data).

```
In [49]: ident_df = la.get_identifiability_dataframe()  # sing val trunc defaults to pst.nnz_ob
```

```
In [50]: ident_df.sort_values(by="ident",ascending=False).iloc[0:20].loc[:,"ident"].plot(kind="
```

```
Out[50]: <matplotlib.axes._subplots.AxesSubplot at 0x1b50bc41518>
```



Note similarity with some of the earlier parameter contribution to forecast uncertainty results

```
In [51]: css = la.get_par_css_dataframe()
         css.head()
```

```
Out[51]:                pest_css  hill_css
         dc0000390005   0.000357       0.0
         dc0000390006   0.000357       0.0
         dc0000390007   0.000437       0.0
         dc0000390008   0.000437       0.0
         dc0000390009   0.000437       0.0
```

```
In [52]: css.sort_values(by="pest_css",ascending=False)
```

```
Out[52]:                 pest_css  hill_css
         flow_0001      16.373237       0.0
         strt6_cn        9.475158       0.0
         rech4_cn        6.472873       0.0
         hk8_cn          3.219146       0.0
         welflux_000     1.546175       0.0
         hk6_cn          0.596127       0.0
         rech008         0.473190       0.0
```

wf0200290006	0.441978	0.0
rech004	0.436865	0.0
rech000	0.409430	0.0
rech011	0.409349	0.0
rech001	0.373742	0.0
rech014	0.350733	0.0
hk202	0.343158	0.0
rech017	0.342363	0.0
wf0200260010	0.324295	0.0
rech021	0.322725	0.0
rech005	0.311130	0.0
rech002	0.310894	0.0
hk229	0.301903	0.0
hk203	0.297866	0.0
rech025	0.285918	0.0
hk206	0.284214	0.0
rech022	0.283196	0.0
hk230	0.251524	0.0
rech029	0.251069	0.0
rech018	0.248949	0.0
rech026	0.246733	0.0
wf0200340012	0.246151	0.0
rech006	0.240714	0.0
...	...	...
ss128	0.000000	0.0
ss126	0.000000	0.0
ss224	0.000000	0.0
ss125	0.000000	0.0
ss124	0.000000	0.0
ss123	0.000000	0.0
ss122	0.000000	0.0
ss121	0.000000	0.0
ss120	0.000000	0.0
ss119	0.000000	0.0
ss203	0.000000	0.0
ss204	0.000000	0.0
ss205	0.000000	0.0
ss206	0.000000	0.0
ss223	0.000000	0.0
ss222	0.000000	0.0
ss221	0.000000	0.0
ss220	0.000000	0.0
ss219	0.000000	0.0
ss218	0.000000	0.0
ss217	0.000000	0.0
ss216	0.000000	0.0
ss215	0.000000	0.0
ss214	0.000000	0.0

```
ss213          0.000000       0.0
ss212          0.000000       0.0
ss210          0.000000       0.0
ss209          0.000000       0.0
ss207          0.000000       0.0
ss020          0.000000       0.0

[527 rows x 2 columns]
```