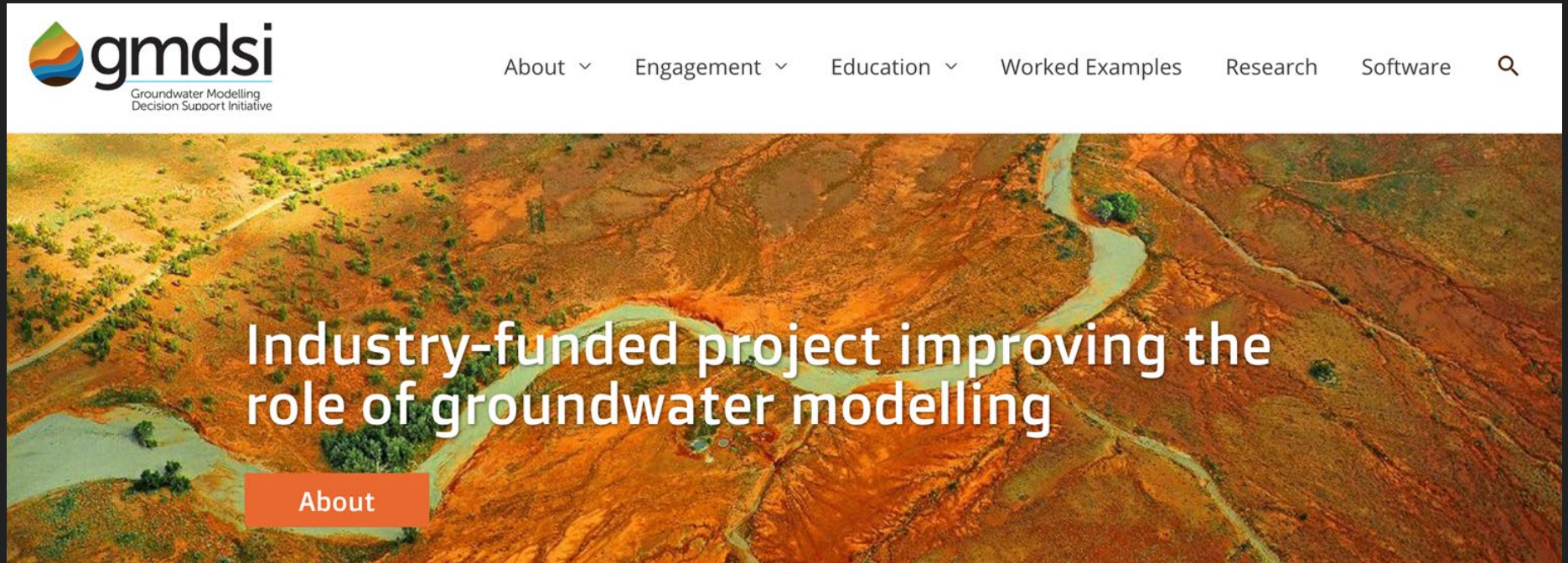
# A whirlwind tour of PEST++ (v5) and pyEMU (v1)

Jeremy White and Mike Fienen

# Thanks to GMDSI!

https://gmdsi.org/



# Thanks to contributors!

- **PEST++**
  - Dave Welter, John Doherty, Randy Hunt, Mike Fienen, Wes Kitlasten, Matt Knowling, Mike Towes, Ayman Alzraiee, Zak Stanko
- **pyEMU**
  - Mike Fienen, Wes Kitlasten, Matt Knowling, Brioch Hemmings, Ayman Alzraiee, Otis Rea

**Always looking for contributors!**

# Outline

1. PEST++ overview
2. pyEMU overview
3. PEST++ worked example
4. pyEMU worked example
5. Looking forward

# Why PEST and PEST++?

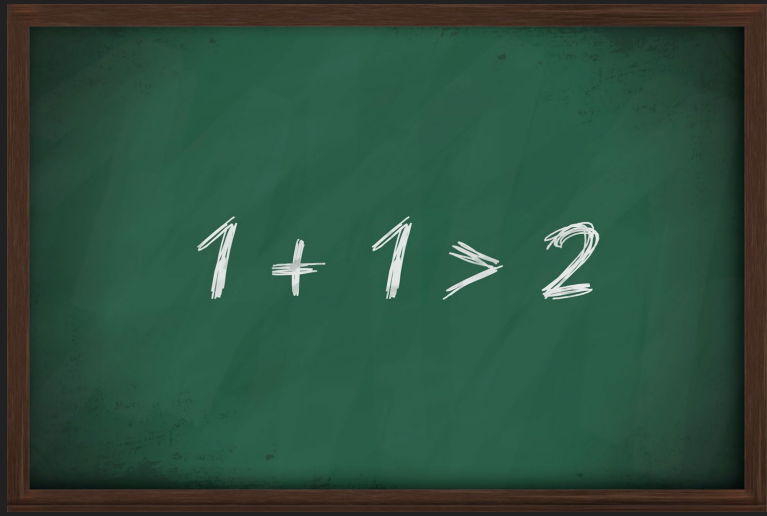
- For decision support, quantifying uncertainty (UQ) and reducing uncertainty thru data assimilation (DA) is fundamental
- This means we need UQ and DA tools that work with a range of models!
- **See previous GMDSI webinars for more theory and concepts related to this topic!**


Shmueli G. (2010) To explain or to predict?. Statistical science

# PEST++

## Design Philosophy

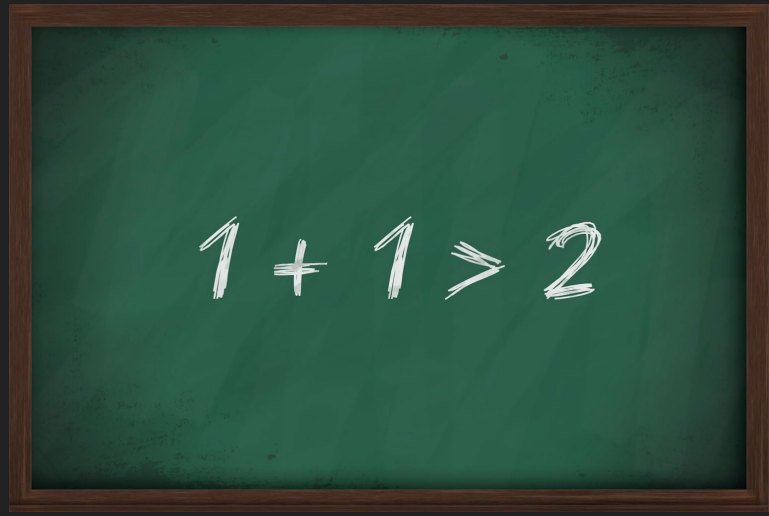
- Compliment the capabilities of PEST
  - Automate/combining workflows
  - New/different algorithms
- Focus on uncertainty and risk



# PEST++

## Design Philosophy

- C++11
- Statically compiled - “stand alone”
- PC, linux, mac
- Serial and parallel run mgr in all tools
- Can be compiled/debugged/profiled with FREE visual studio and MSVC



# PEST++

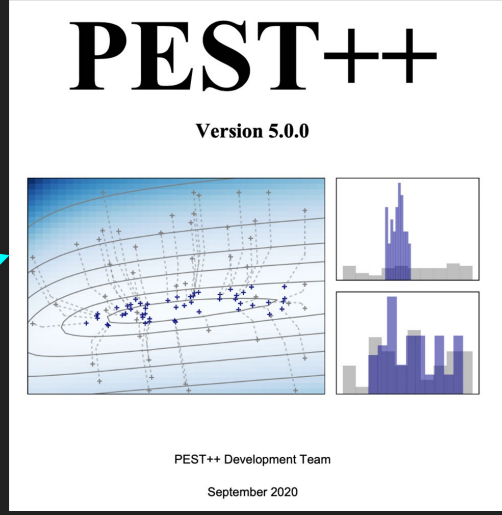
## Overview of  (current) capabilities

- Global Sensitivity Analyses
  - Diagnostics and “plumbing problems”
- Data Assimilation and Uncertainty Analyses
  - Bayesian parameter conditioning
- Management Optimization Under Uncertainty
  - Risk-based optimal resource management
- Generic parallel run management
  - Design of experiments, emulator training, etc

# PEST++

## Codes, documentation and support

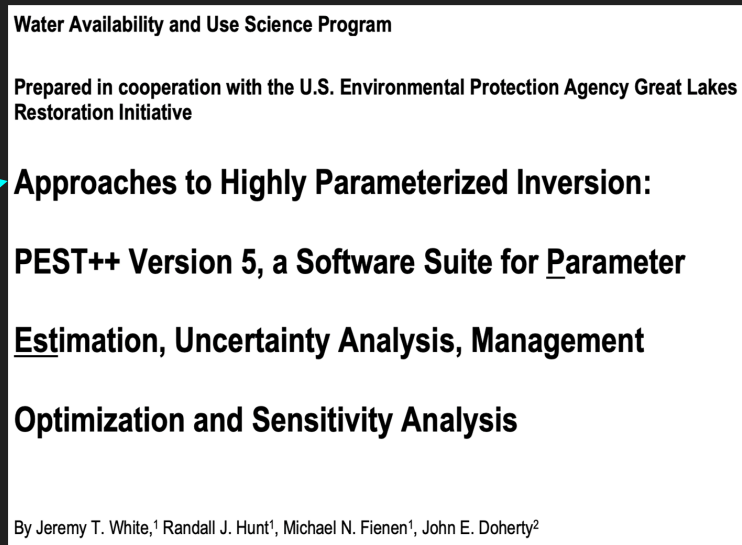
- **https://github.com/usgs/pestpp**
  - Precompiled binaries
  - source + VS solution + make/cmake
  - Users manual
  - GH “issues” for support
    - Feature requests
    - Bug reports
    - General complaining
- PEST++ V5 USGS report
  - “in press”


PEST++

Version 5.0.0

PEST++ Development Team

September 2020


Water Availability and Use Science Program

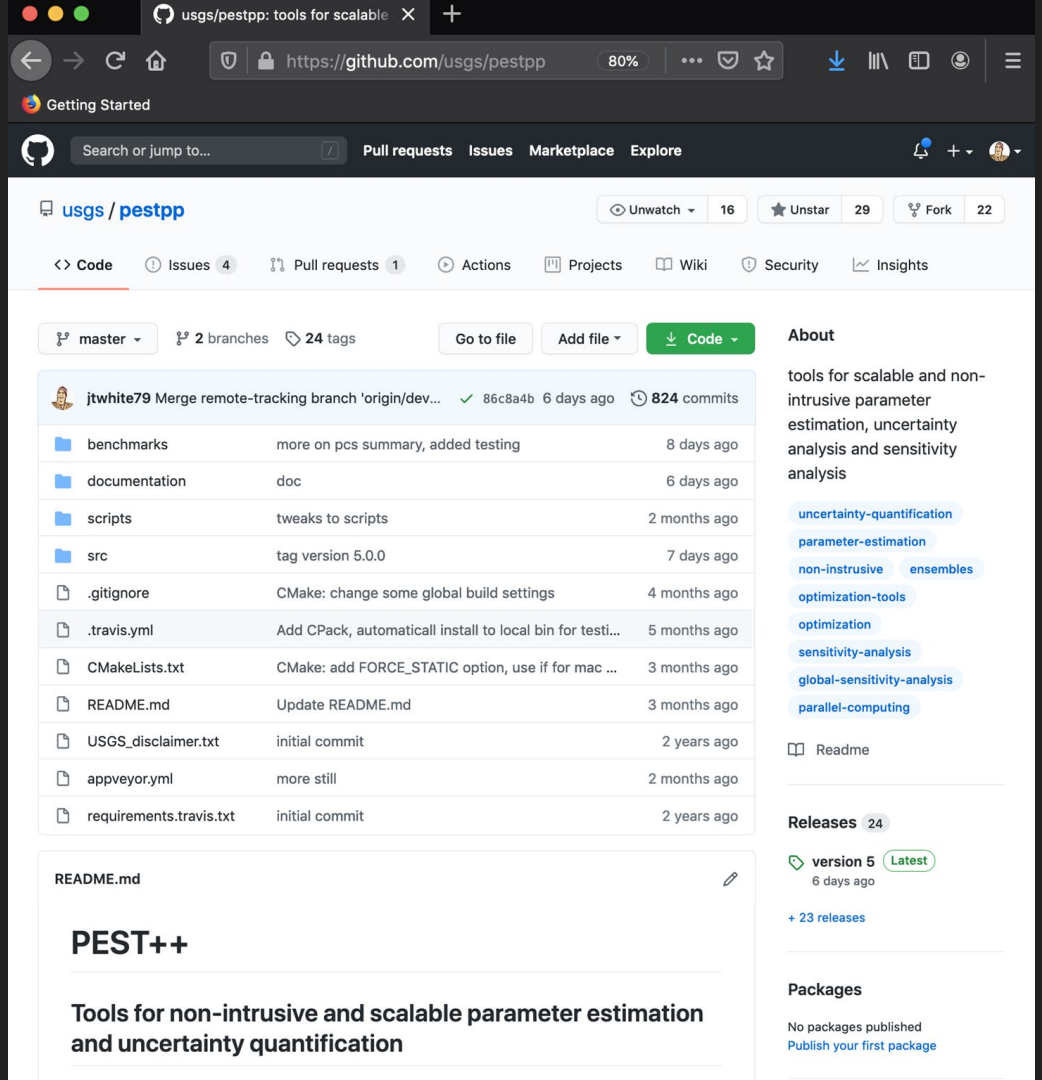
Prepared in cooperation with the U.S. Environmental Protection Agency Great Lakes Restoration Initiative

Approaches to Highly Parameterized Inversion:

PEST++ Version 5, a Software Suite for Parameter Estimation, Uncertainty Analysis, Management Optimization and Sensitivity Analysis

By Jeremy T. White,[1] Randall J. Hunt[1], Michael N. Fienen[1], John E. Doherty[2]

https://github.com/usgs/pestpp



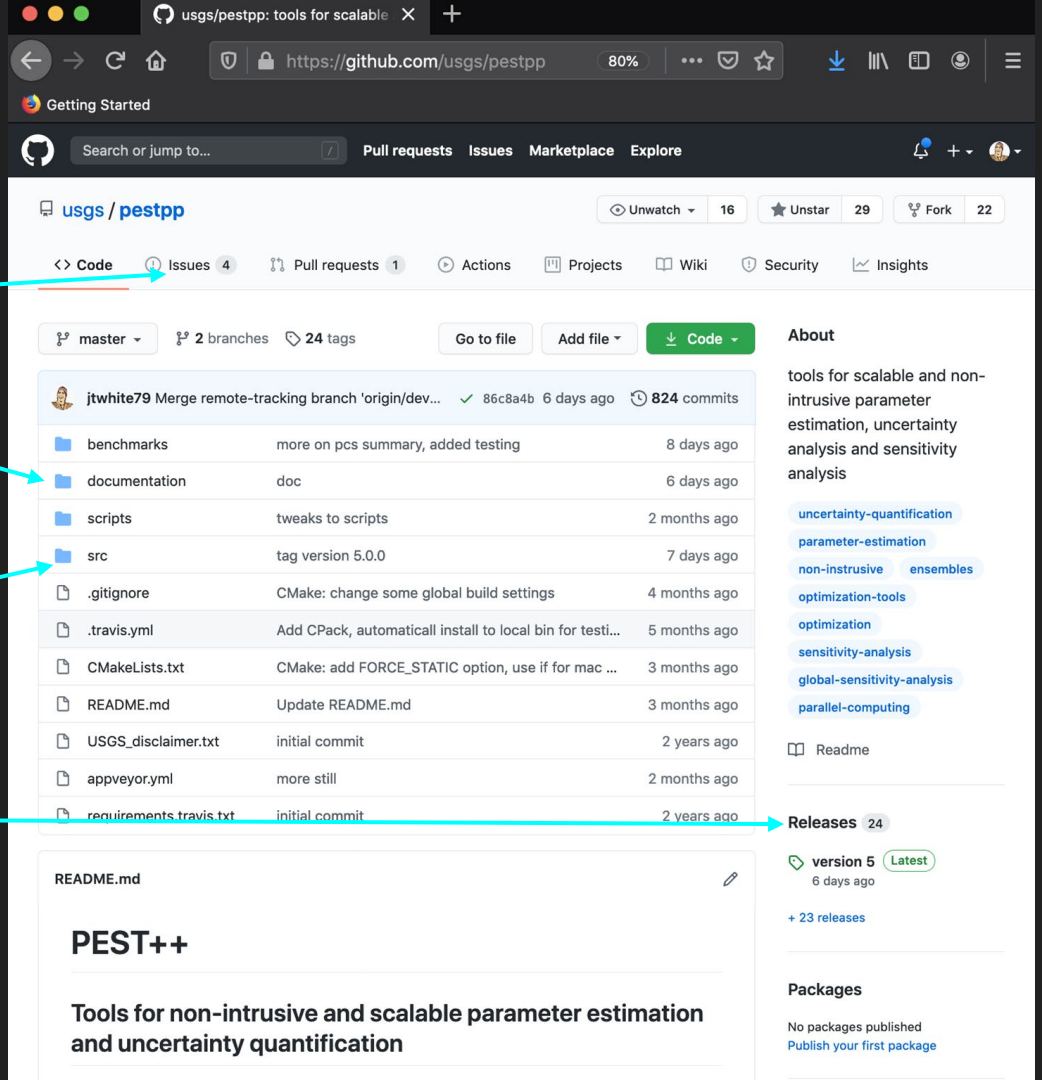
https://github.com/usgs/pestpp

issues

Users manual
(thanks J Doh!)

Source code and
VS solution

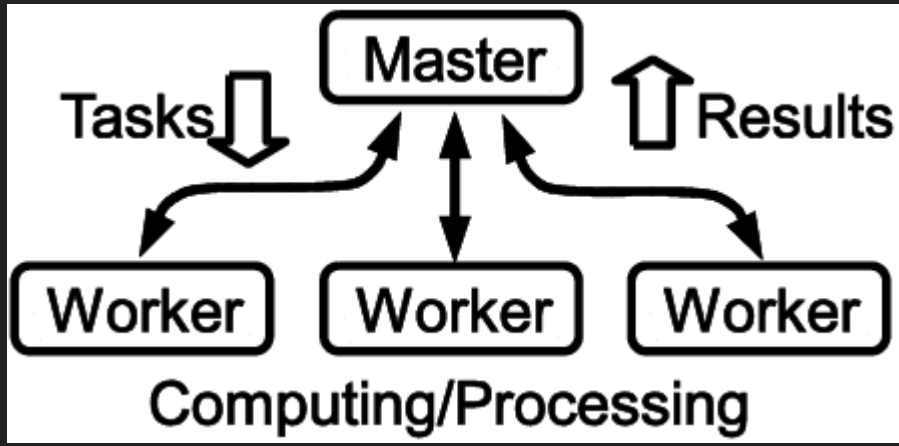
Pre-compiled
binaries
(PC and linux)



# PEST++

## Run management

- All pestpp-xxx tools have these!
- Serial run mgr
  - One run at a time
  - >>>pestpp-xxx my.pst
- Parallel run mgr
  - Master-worker concept
  - tcp/ip socket programming
  - Master: >>>pestpp-xxx my.pst /h :4004
  - Worker: >>>pestpp-xxx my.pst /h 111.222.333:4004
  - Multithreaded workers
    - Comms during forward run
    - Exception handling



# PEST++

## Using PEST(_HP) vs PEST++

### PEST(_HP)

- Control file
- Template files
- Instruction files
- Jacobian files
- Uncertainty files
- etc
- >>>pest_hp my.pst

### PEST++

- Control file
- Template files
- Instruction files
- Jacobian files
- Uncertainty files
- etc
- >>>pestpp_xxx my.pst

# PEST++

## Using PEST(_HP) vs PEST++

**If your model is setup for PEST(_HP),**

**your model is also setup for PEST++**

# PEST++

## Version 5: PURELY OPTIONAL variations from PEST(_HP)

- “++” control file args to modify internal defaults
  - PEST(_HP) ignores these
  - See user’s manual!
- Name lengths
  - 200-char parameter and observation name lengths
  - Automatic handling in PEST++
- Enhanced  (“version 2”) control file

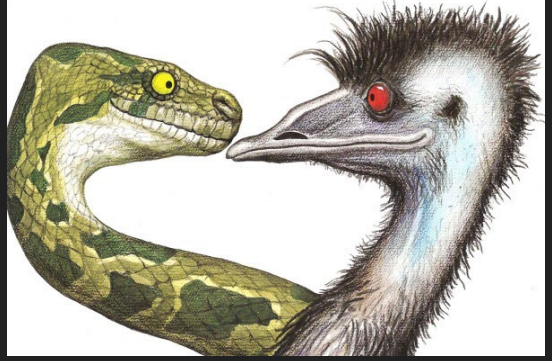
# Version 2 control file

- Allows a wider range of algorithms
- Keyword-value pairs
- Defaults for all algorithmic variables
- External CSV files
- See user's manual for specifications

Control file with 2M pars and 6M obs

```
pcf version=2
* control data keyword
noptmax   0
ies_par_en prior.jcb
* parameter groups external
clas.pargrp_data.csv
* parameter data external
clas.par_data.csv
* observation data external
clas.obs_data.csv
* model command line
python forward_run.py
* model input external
clas.tplfile_data.csv
* model output external
clas.insfile_data.csv
```

# pyEMU



- https://github.com/pypest/pyemu
- Python interface/wrapper for PEST and PEST++
- “All things PEST and PEST++”
- programmatic PEST(++) setup and processing

White and others (2016). A python framework for environmental model uncertainty analysis

# pyEMU

## Why pyEMU

- PEST (and the PEST utilities) and PEST++ are complex tools
  - Many opportunities for hardship!
- Just like the forward model, small “changes” to inputs can cause large changes in outputs
- pyEMU can “automate” the implementation of PEST and PEST++ analyses
  - Decreased cognitive load
  - Increased efficiency
  - Reproducibility

# pyEMU + PEST(++)

## Workflow automation - efficiency and reproducibility

- Combining the efficiency of ensemble methods with the automation in pyEMU
- No painful tradeoffs between number of parameters and computation
- Documenting decisions about models all through the process
- Much less despair for “redos”
- Minimum Viable Product

Hydrol. Earth Syst. Sci., 20, 3739–3743, 2016
www.hydrol-earth-syst-sci.net/20/3739/2016/
doi:10.5194/hess-20-3739-2016

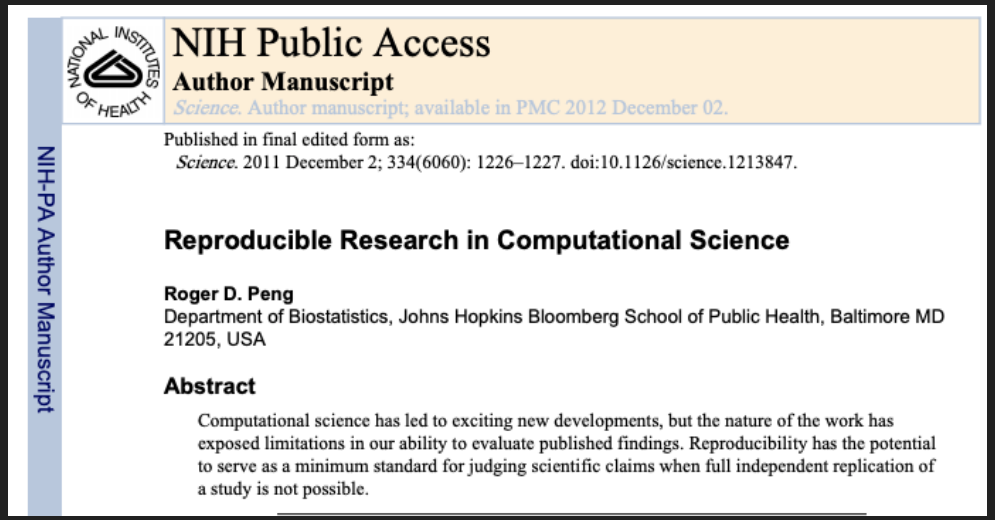

Hydrology and Earth System Sciences
Open Access
EGU

**HESS Opinions: Repeatable research: what hydrologists can learn from the Duke cancer research scandal**

**Michael N. Fienen[1] and Mark Bakker[2]**
[1]US Geological Survey Wisconsin Water Science Center, Middleton, Wisconsin, USA
[2]Water Resources Section, Faculty of Civil Engineering and Geosciences, Delft University of Technology, Delft, the Netherlands

*Correspondence to:* Michael N. Fienen (mnfienen@usgs.gov)

Received: 6 May 2016 – Published in Hydrol. Earth Syst. Sci. Discuss.: 20 May 2016
Revised: 25 August 2016 – Accepted: 31 August 2016 – Published: 12 September 2016







Published in final edited form as:
*Science*. 2011 December 2; 334(6060): 1226–1227. doi:10.1126/science.1213847.

**Reproducible Research in Computational Science**

**Roger D. Peng**
Department of Biostatistics, Johns Hopkins Bloomberg School of Public Health, Baltimore MD 21205, USA

**Abstract**

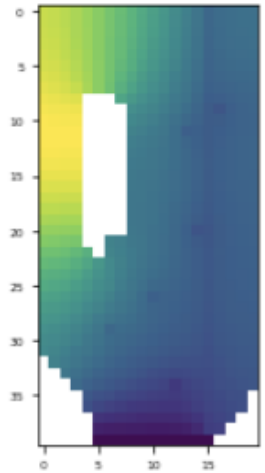
Computational science has led to exciting new developments, but the nature of the work has exposed limitations in our ability to evaluate published findings. Reproducibility has the potential to serve as a minimum standard for judging scientific claims when full independent replication of a study is not possible.

# pyEMU - with Jupyter Notebooks

Let's quickly viz the model top just to remind us of what we are dealing with:

```
id_arr = np.loadtxt(os.path.join(org_model_ws,"freyberg6.dis_idomain_layer3.txt"))
top_arr = np.loadtxt(os.path.join(org_model_ws,"freyberg6.dis_top.txt"))
top_arr[id_arr==0] = np.nan
plt.imshow(top_arr)
```

```
<matplotlib.image.AxesImage at 0x141baf390>
```

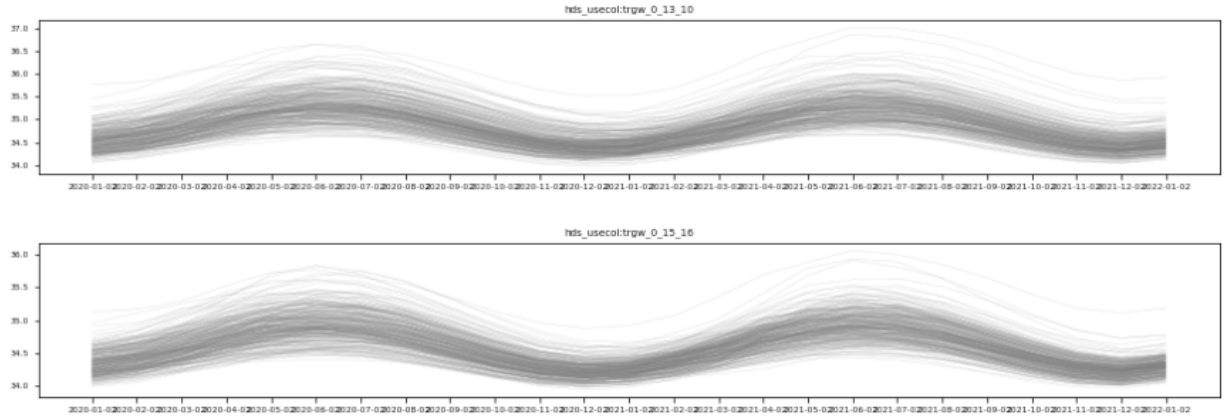



jupyter

# pyEMU - with Jupyter Notebooks

In [10]:

```
pst.try_parse_name_metadata()
#pst.observation_data.loc[:,"time"] = pst.observation_data.time.apply(np.float)
obs = pst.observation_data
for obs_group in pst.obs_groups:
    obs_g = obs.loc[obs.obgnme==obs_group,:].copy()
    obs_g.sort_values(by="datetime",inplace=True)
    obs_g_df = obs_df.loc[:,obs_g.obsnme]
    fig,ax = plt.subplots(1,1,figsize=(15,2))
    [ax.plot(obs_g.datetime,obs_g_df.loc[i,obs_g.obsnme],color='0.5',alpha=0.5,lw=0.2) for
    ax.set_title(obs_group)
    plt.show()


```



# pyEMU

## Design philosophy

- “This should be easier”
  - mimic the Pandas style
- OOP, yeah you know me!
- Inclusive and cooperative development
  - A trick to convince users to help!

# pyEMU

## Overview of capabilities

- “Labeled” linear algebra
  - Less opportunities for hardship!
- Heaps of FOSM
  - PREDUNC PREDVAR, PNULPAR, etc
  - Dataworth
- Interacting with PEST Control Files
  - Using Pandas.Dataframe
- Ensembles
  - parameters, observations (noise)
  - Null-space projection
  - Bayes-linear Monte Carlo
  - Based on Pandas.Dataframe

# pyEMU

## Overview of capabilities

- I/O for all PEST-style files
  - pst, par, res, jcb/jco, mat, vec, unc, etc
- Pilot points and geostats
- Prior covariance matrix and Prior ensemble construction
- Plotting
- Programmatic interface construction

# pyEMU

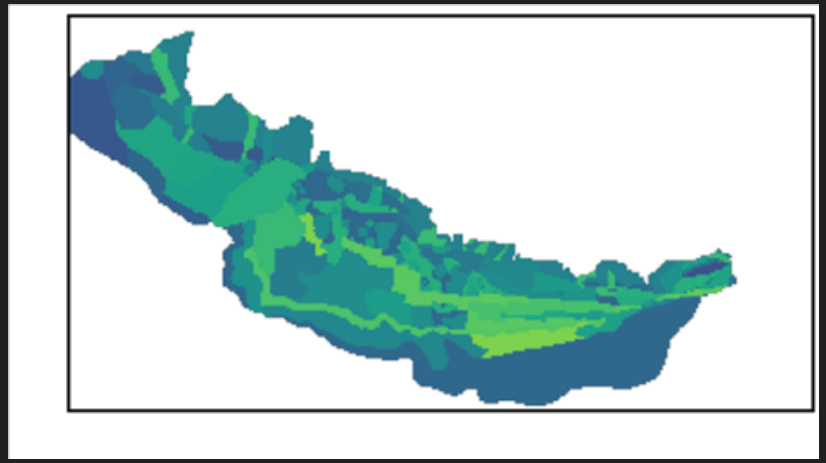
## PstFrom: overview

- **Brioch Hemmings (GNS Science)**
- Programmatic construction of high-dimensional PEST(++) interface
  - Full-meal deal!
- Array-style (2-D) and list-style (tabular) input and output files
- Multiplier parameters
  - Broadcasting across layers and/or time steps
- Nested-spatial and temporal scales of parameters
  - Grid-scale, pilot points, zones, constants
- Geostatistical Prior cov matrix and/or ensembles

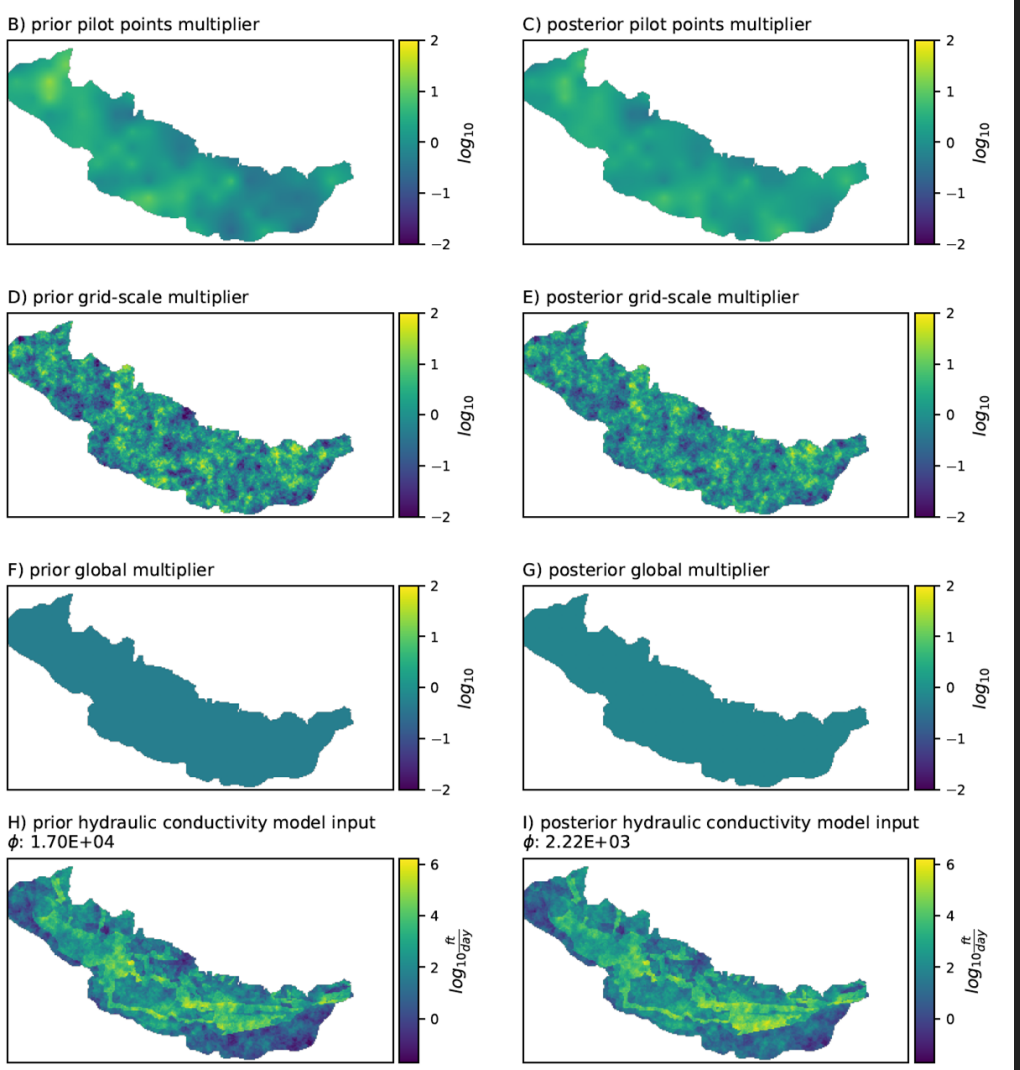
# pyEMU

**PstFrom: multiscale parameterization**

- Preserve existing model inputs as a “full-dimensional” Prior mean
- Estimate “spread” around the mean
- Partition uncertainty according to scale
- Some help in identifying parameter compensation



White and others (2020). Towards reproducible environmental modeling for decision support; A worked example



# pyEMU

**PstFrom: prior covariance matrix and ensemble generation**

- Using minimal spatiotemporal information, construct geostatistical Prior quantities
- Assumes correlation only within parameter groups
- Currently unconditioned realizations only*
    - Spectral simulation

# PEST++ workflow demonstration

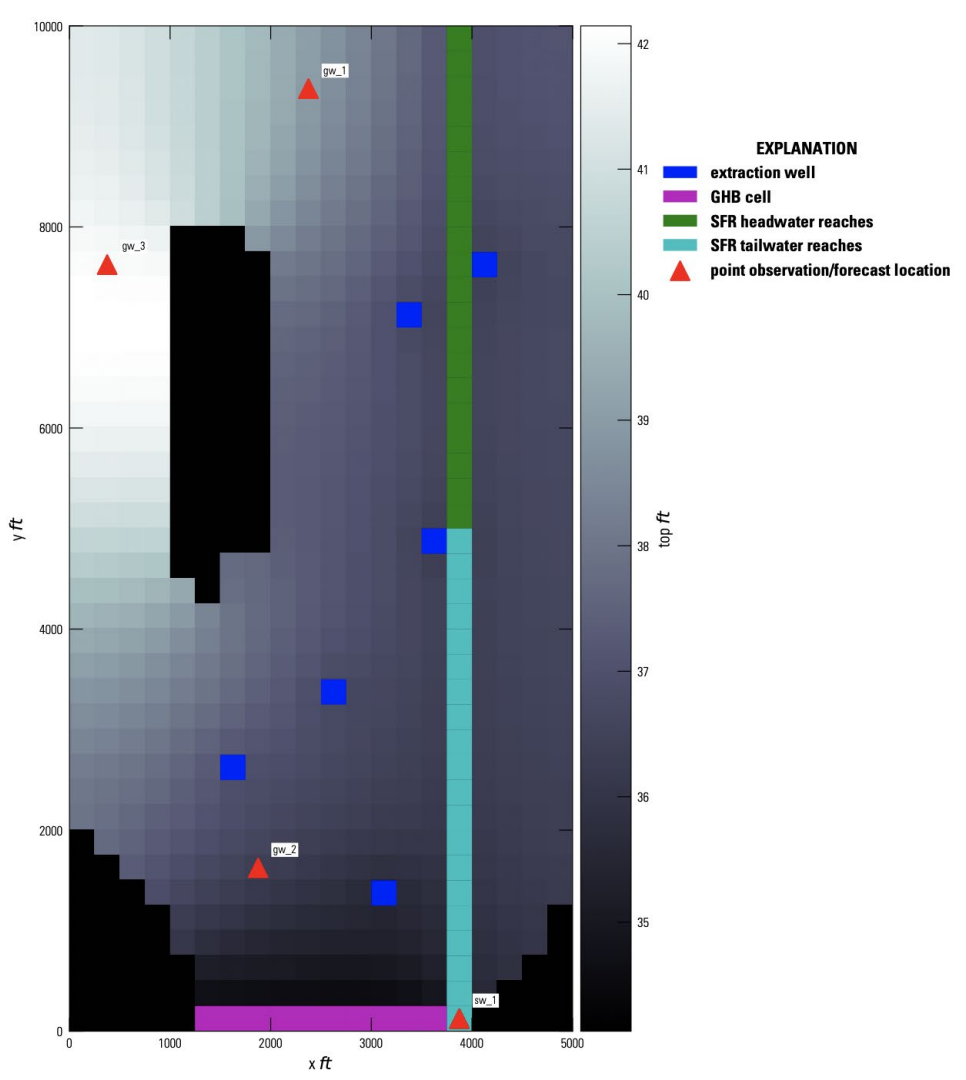
Follows exactly from the PEST++ V5 USGS report!

(that's still in press…)

# The (synthetic) model

## Freyberg history and context

- Freyberg (1988)
- Hunt and others (2019)
- MODFLOW-6
- 3 layers X 40 rows X 20 cols
- 25 stress periods
  - 1 SS
  - 24 monthly transient
- SFR, WEL, RCH, GHB



# Python-less Example Workflow

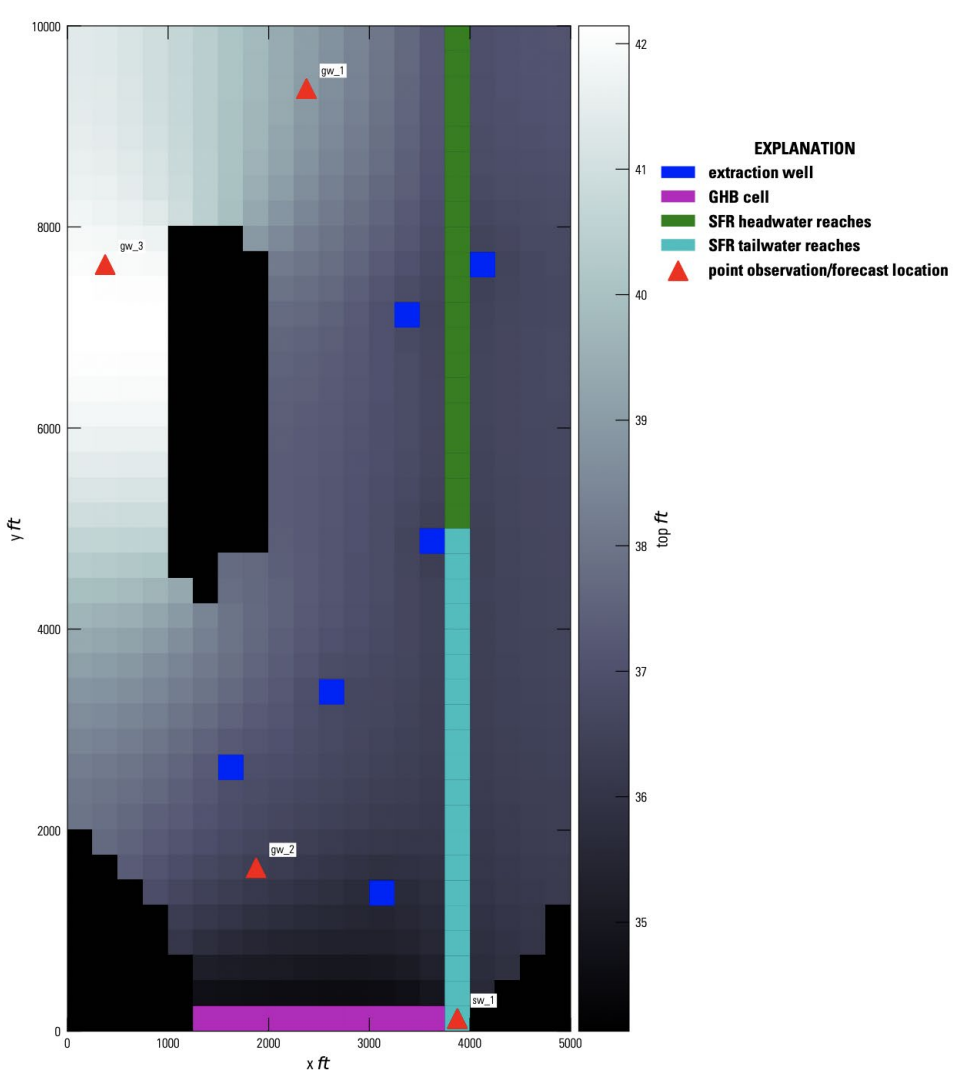
## Following (exactly) the PEST++ Version 5 Report

1. PESTPP-SEN: diagnostics/error checking
2. PESTPP-GLM/PESTPP-IES: uncertainty and data assimilation
3. PESTPP-OPT: risk-based optimal management solutions

# Python-less Example

## The PEST(++) interface

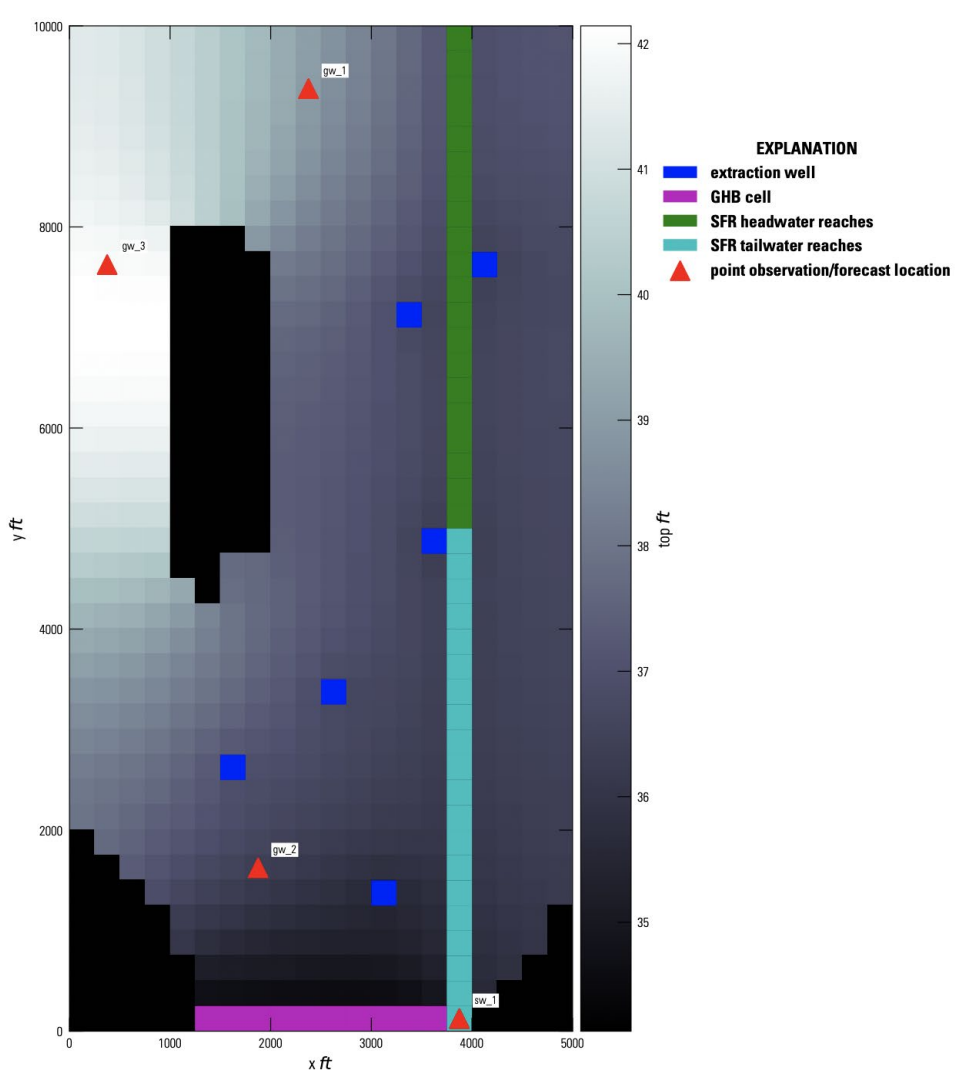
- 8,175 parameters
  - HK, VK, SS, SY, WEL, RCH, SFR, GHB
- Historic Period
  - First 12 transient periods
  - GW levels at gw_1 and gw_2
  - SW flow at sw_1
- Forecast Period
  - Last 12 transient periods



# Python-less Example

## Forecasts

- Tailwater sw-gw exchange at end of historic period
  - solution space
- Headwater sw-gw exchange at the end of the forecast period
  - null space
- Water level at gw_3 at the end of the forecast period
  - solution/null space

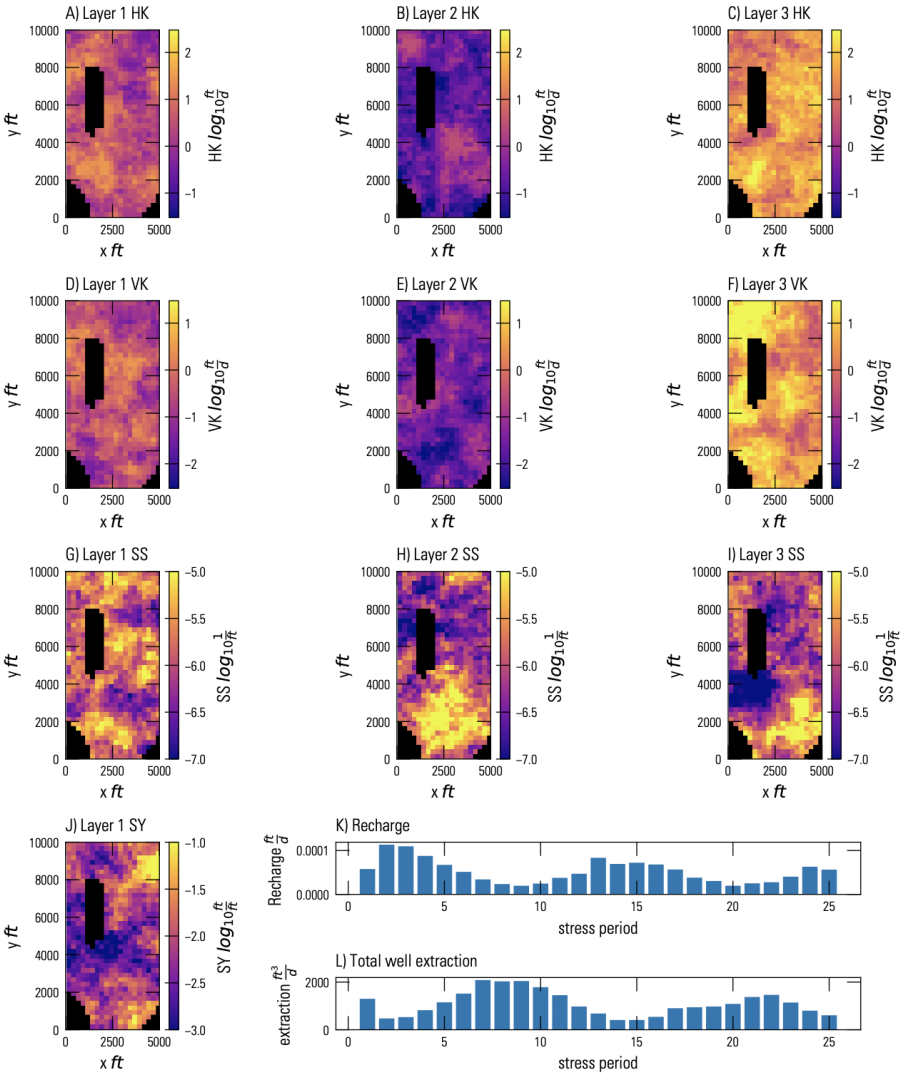


# Very Important Point

**The quantities of interest (e.g. the forecasts) are part of the PEST interface so we can track their value(s) throughout the following analyses.**

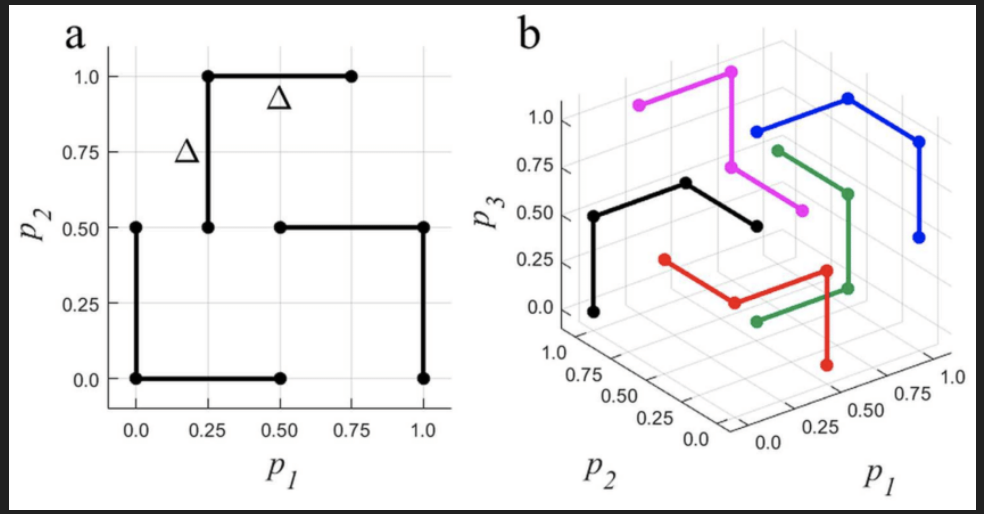
# The “truth”

- From 300 realizations…
  - pyEMU/PEST utils
- 95th percentile tailwater forecast
- A hard forecast to hit!
  - Extreme first moment
  - Small second moment
- Use outputs from this model
  - Historic observations (DA)
    - gw_1, gw_2, and sw_1
  - Forecast “truths”



# PESTPP-SEN: overview

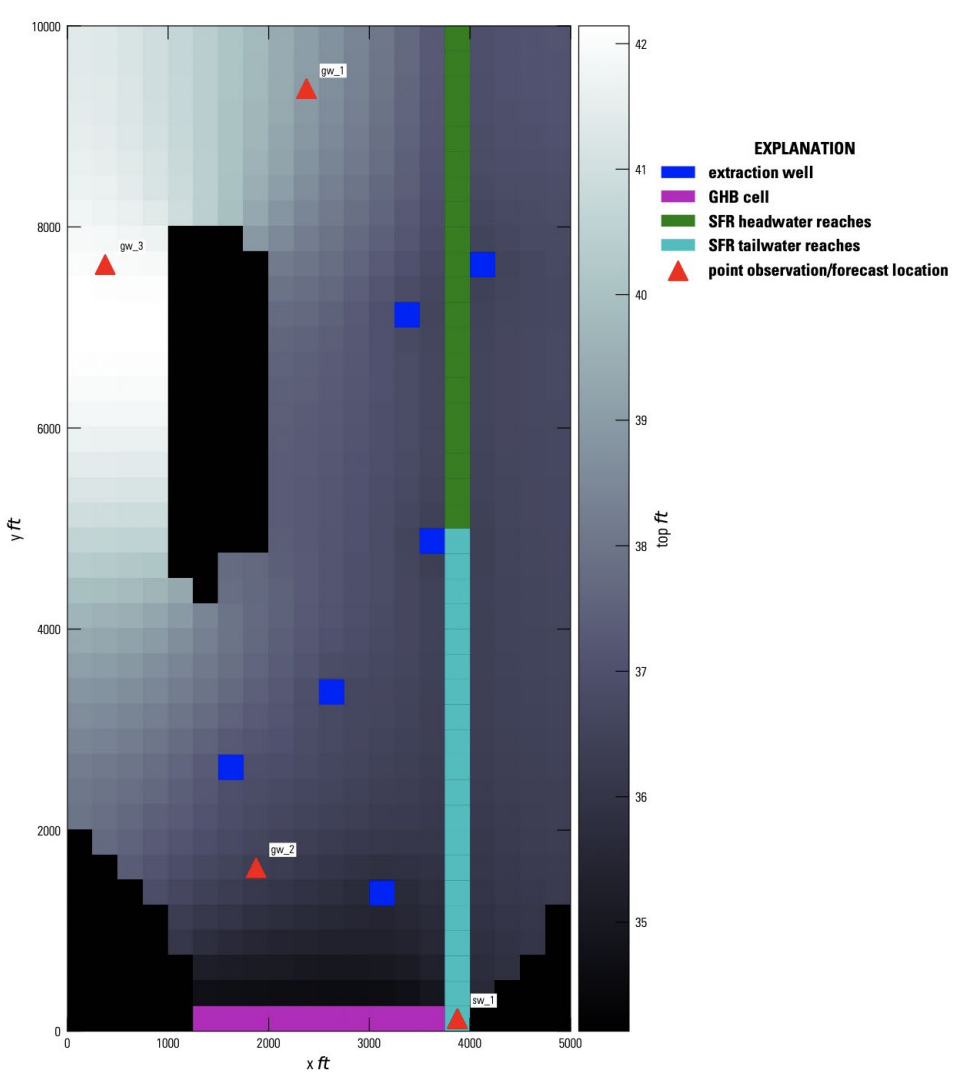
- Morris (and Sobol) global sensitivity analysis
- Morris: “One at a time” method
  - Samples sensitivities across parameter space
- Yields mean and standard deviation of parameter sens to obs in ctl file
- “It just works”: by default, runs =  4 X (num par + base run)



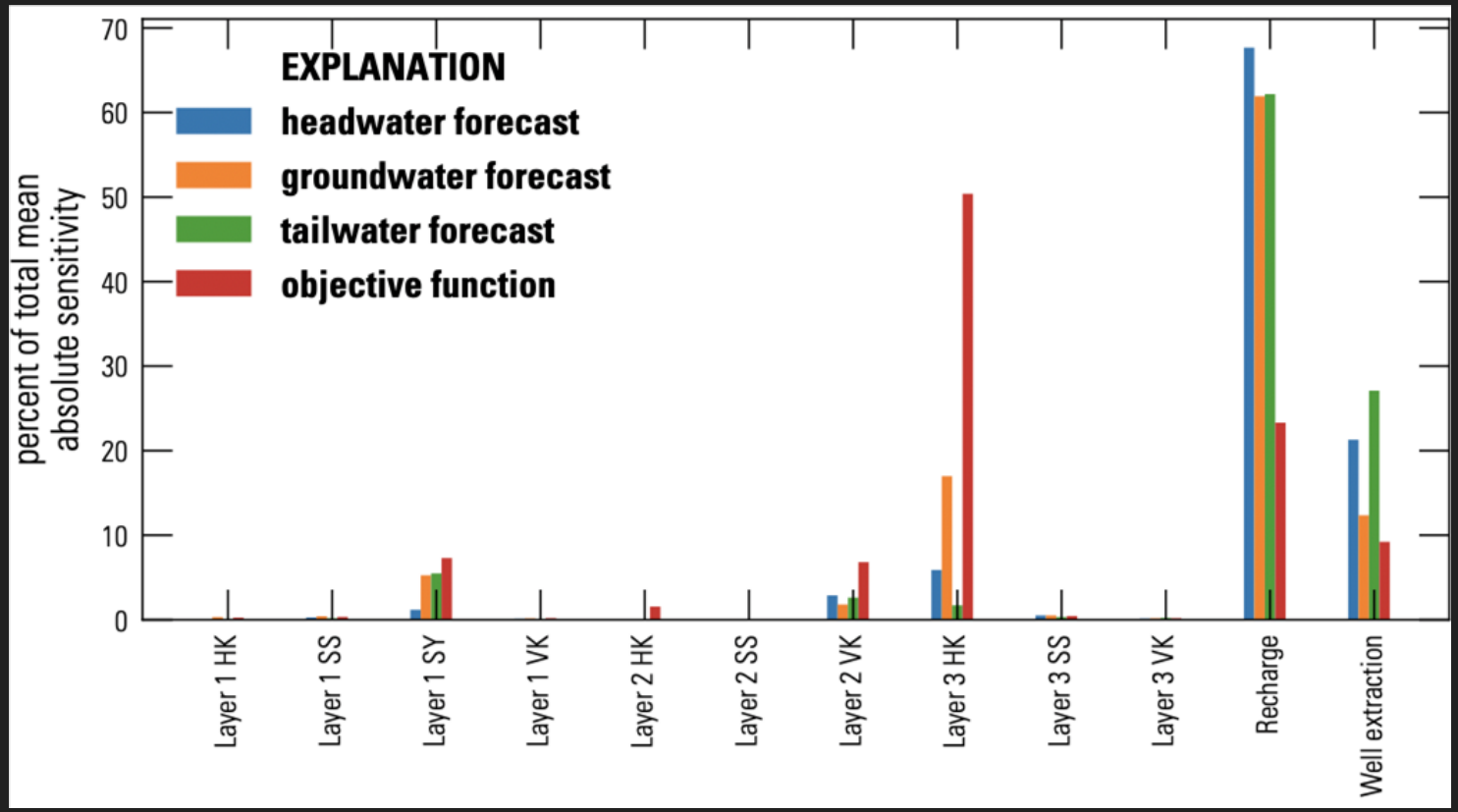

Likhachev, D. V. Parametric sensitivity analysis as an essential ingredient of spectroscopic ellipsometry data modeling: An application of the Morris screening method."

# PESTPP-SEN: setup

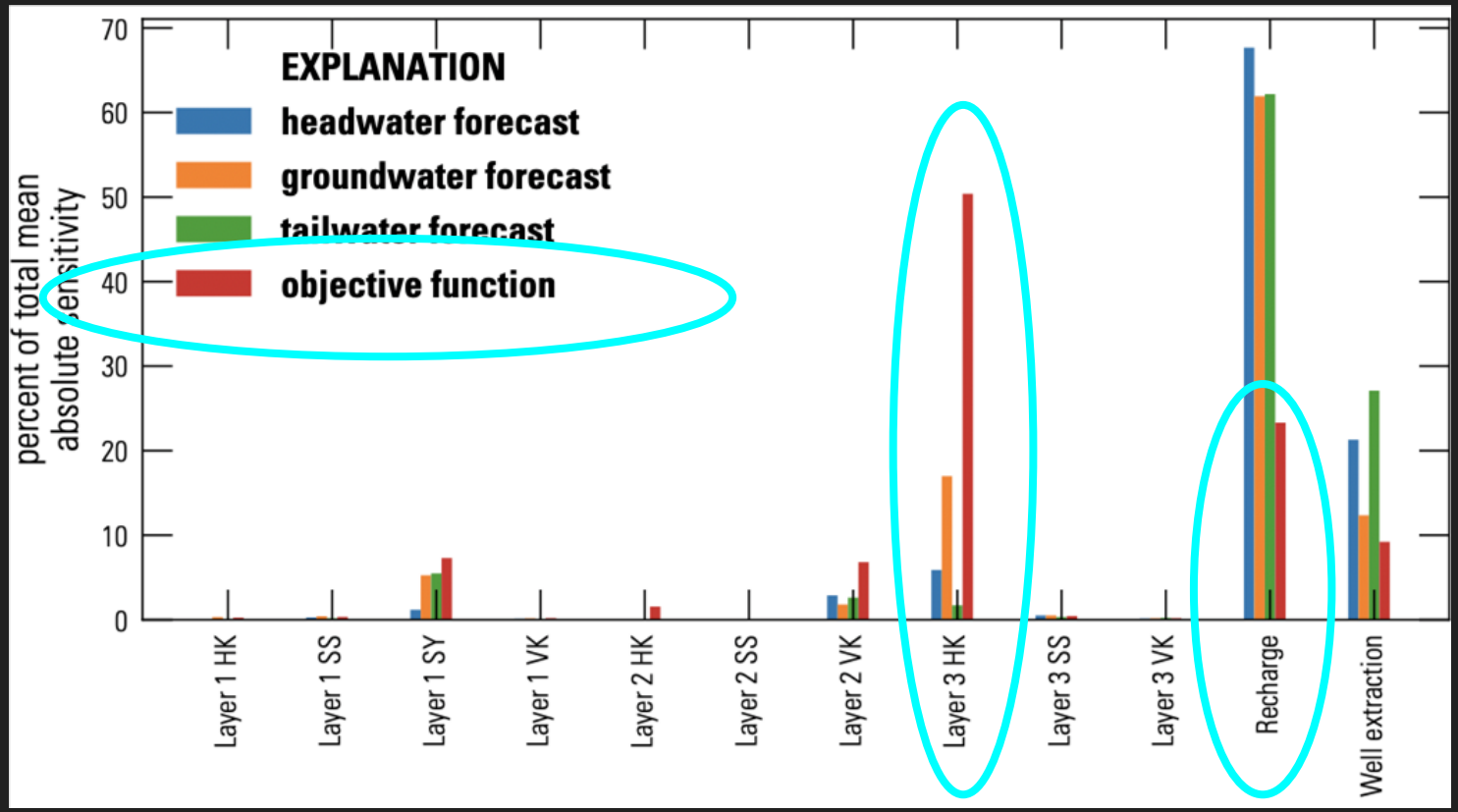
- ++tie_by_group(true) - drastically reduce the number of “adjustable” parameters from 8,175 to 12
- 52 model runs



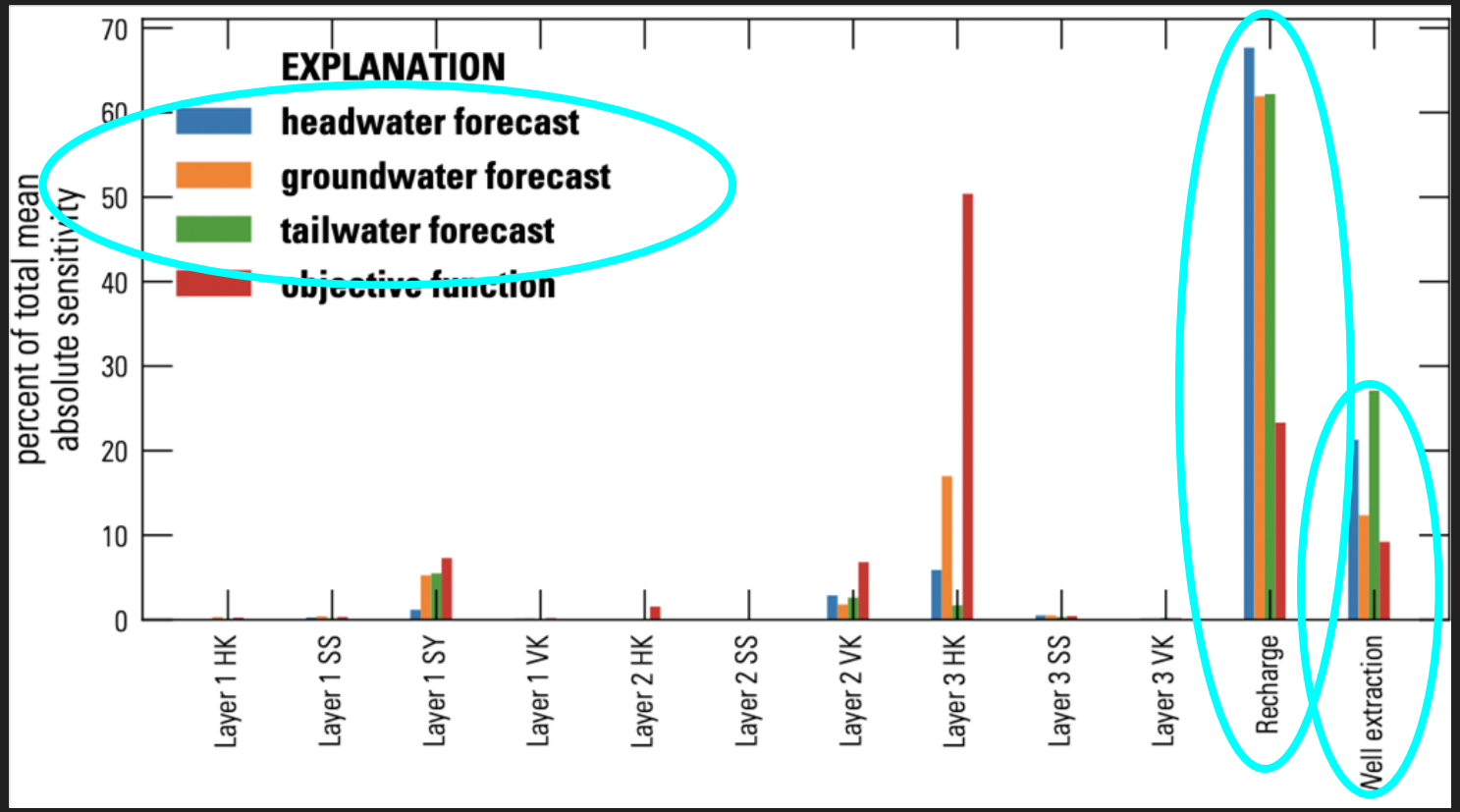
# PESTPP-SEN: results



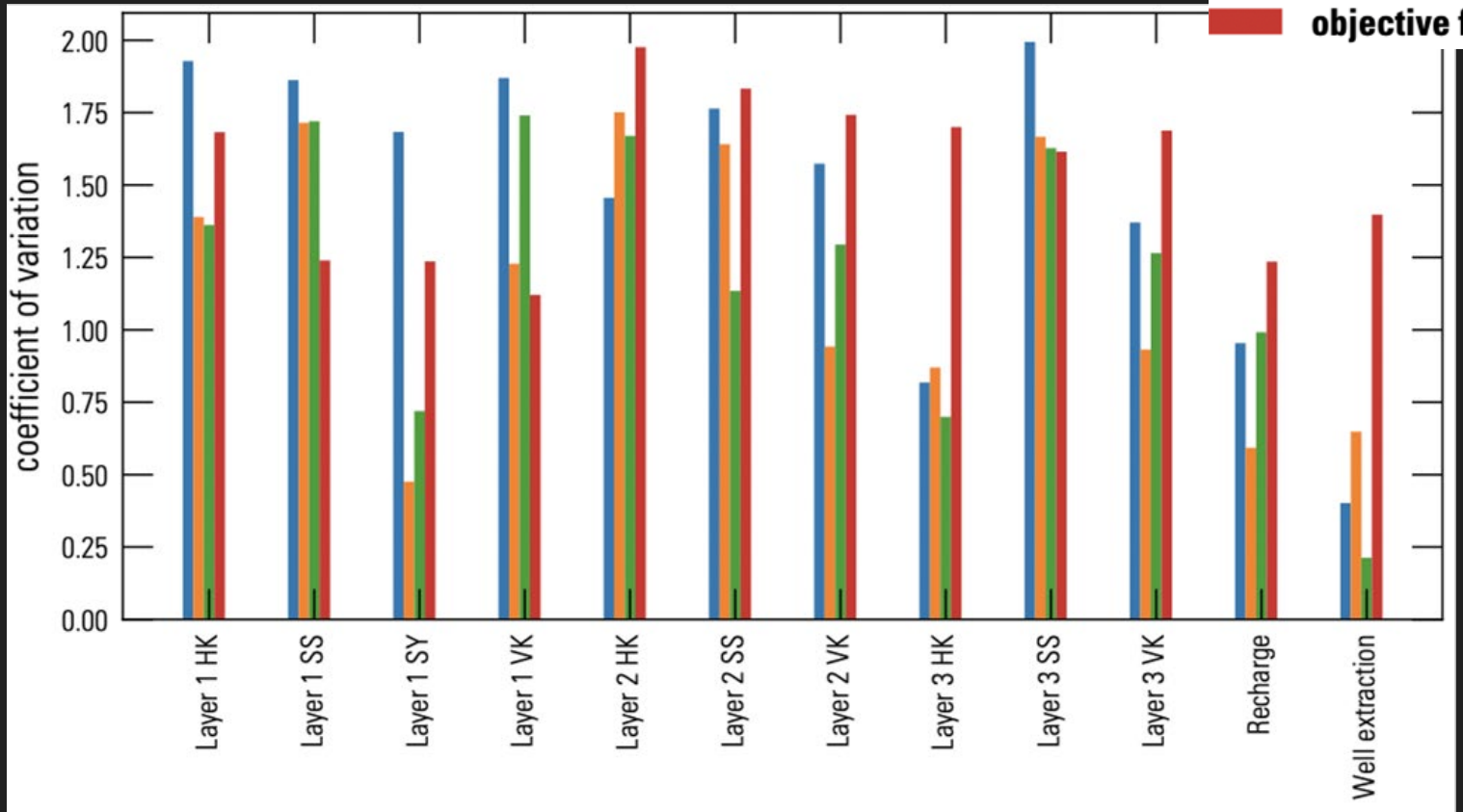
# PESTPP-SEN: results



# PESTPP-SEN: results



# PESTPP-SEN: results



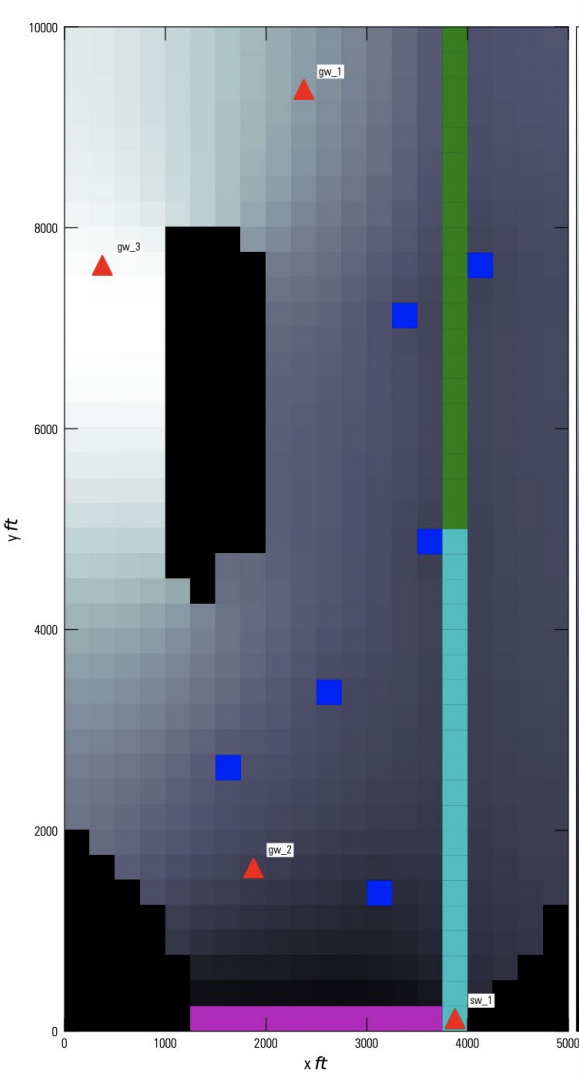
# PESTPP-GLM: overview

- The code formerly known as “pest++”
  - Regularized Gauss-Levenburg-Marquardt
  - Automated “super parameters”/“SVD-Assist”
- Focus on uncertainty
  - Automated posterior FOSM estimation each iteration
    - PREDUNC1 workflow
    - “For free” - no additional runs
  - Automated bayes-linear (aka FOSM-based, linear-assisted) Monte Carlo
    - PREDUNC7 + RANDPAR workflow
    - Draw and run realizations from the posterior covariance matrix

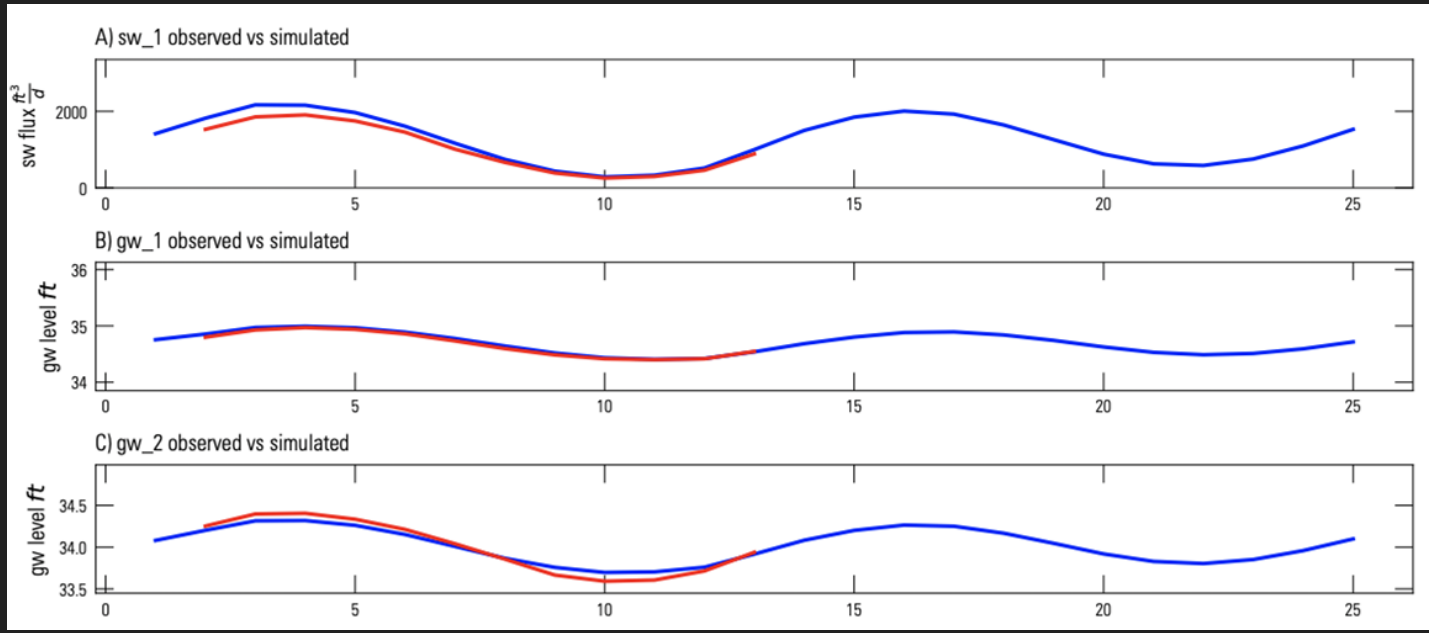
$$\underbrace{\mathbf{p}_\Delta}_{\text{upgrade vector}} = -\Big(\underbrace{\left(\mathbf{J}^T\boldsymbol{\Sigma}_\epsilon^{-1}\mathbf{J}\right)}_{\text{approx Hessian}} + \underbrace{(1+\lambda)\boldsymbol{\Sigma}_\mathbf{p}^{-1}}_{\text{dampening}}\Big)^{-1}\Big(\boldsymbol{\Sigma}_\mathbf{p}^{-1}\underbrace{(\mathbf{p}-\mathbf{p}_0)}_{\text{par change vector}} + \mathbf{J}^T\boldsymbol{\Sigma}_\epsilon^{-1}\underbrace{(\mathbf{d}_{\text{sim}}-\mathbf{d}_{\text{obs}})}_{\text{residual vector}}\Big)$$

# PESTPP-GLM: setup

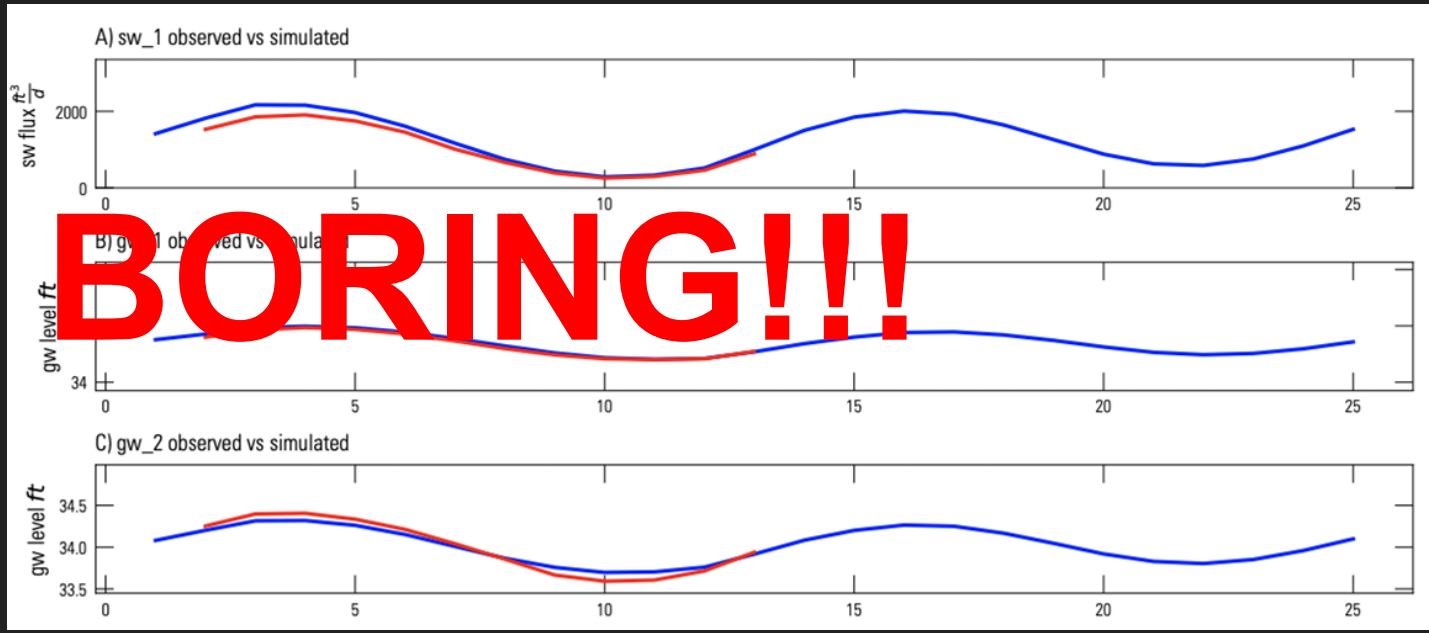
- Tied spatially-distributed parameters into “blocks” or zones of 6 X 6 cells
  - Not ideal compared to pilot points
  - Still express spatial heterogeneity
  - Reduced number of parameters from 8K to 330
- Automated SVD-Assist
  - ++n_iter_base(-1)
  - ++n_iter_super(3)
- ++glm_num_reals(200)
- 682 model runs
  - 331 to fill the full parameter jco
  - 151 for super parameter solution process
  - 200 for posterior Monte Carlo



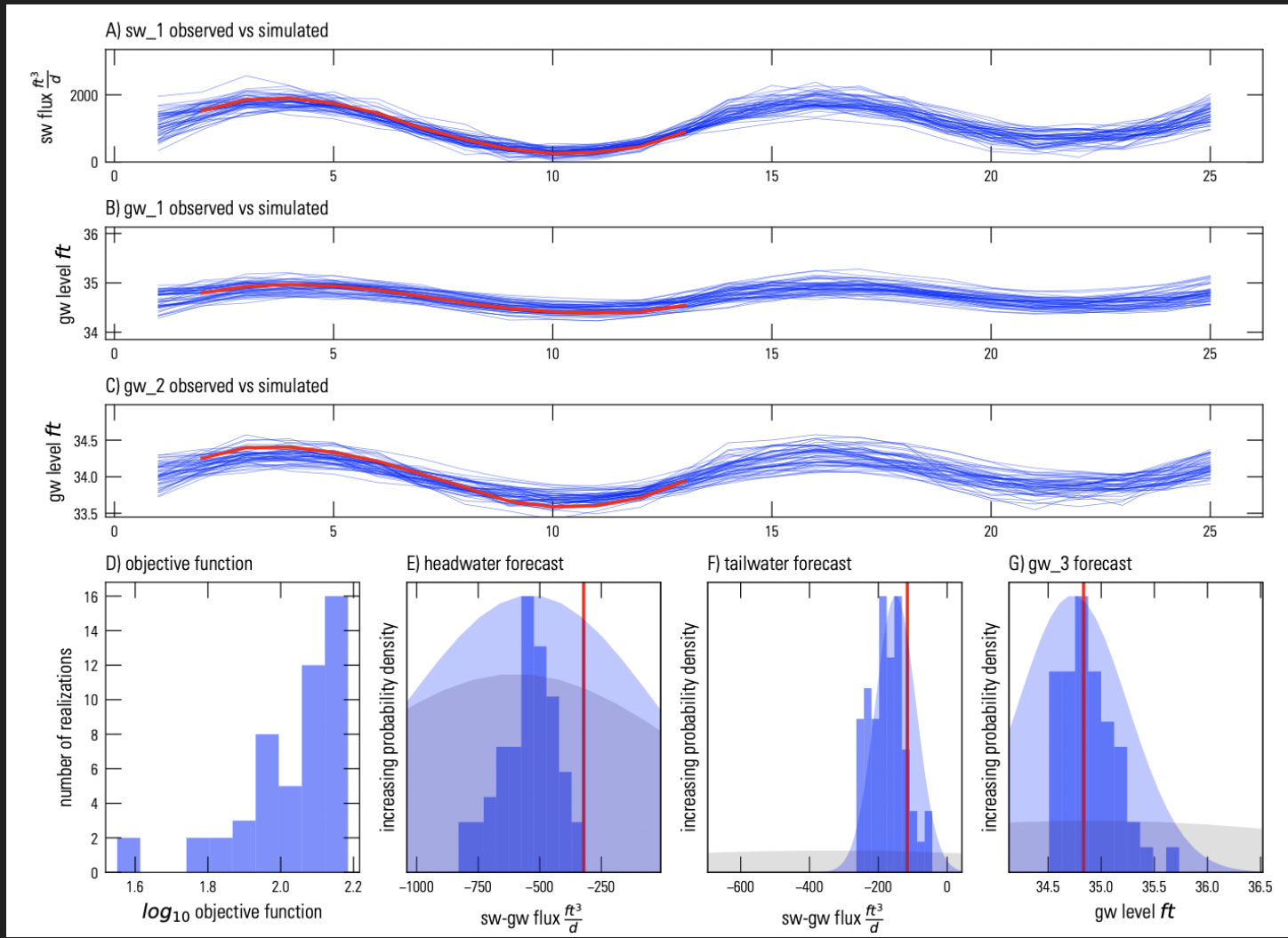
- Red = observed
- Blue = posterior



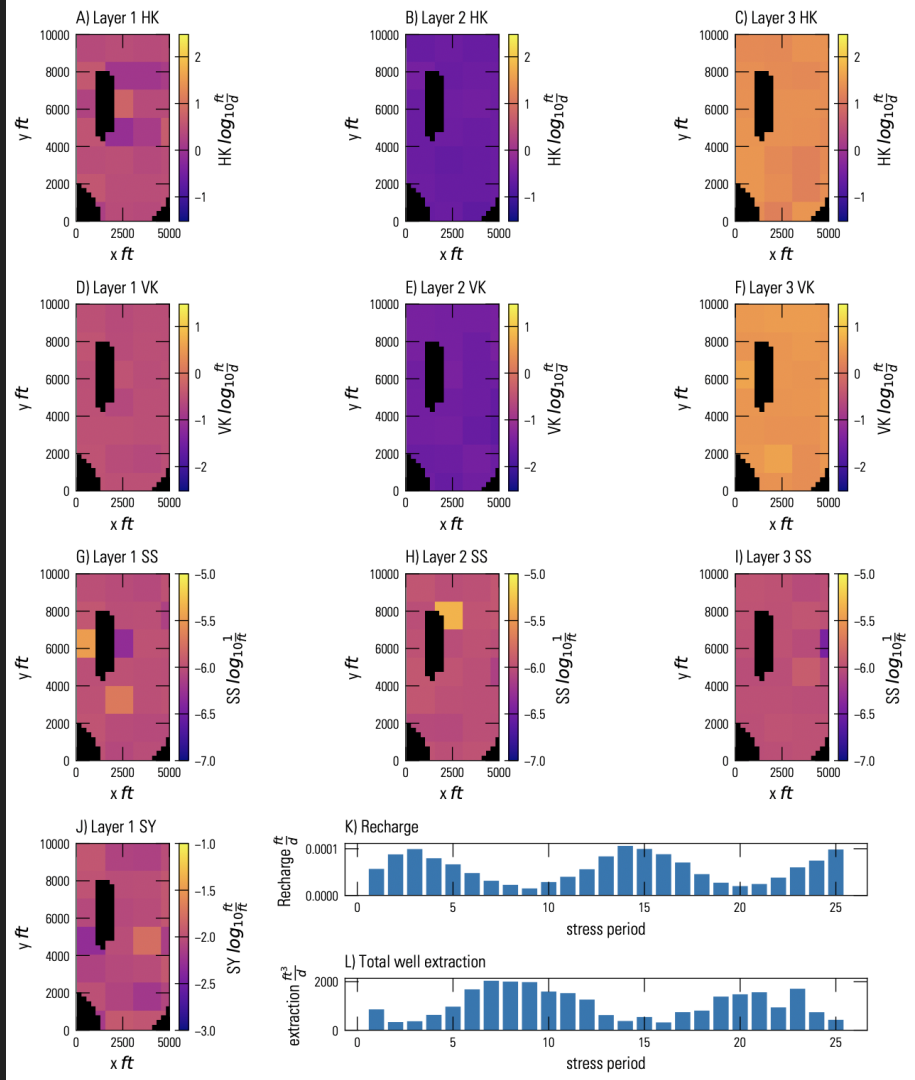
- Red = observed
- Blue = posterior



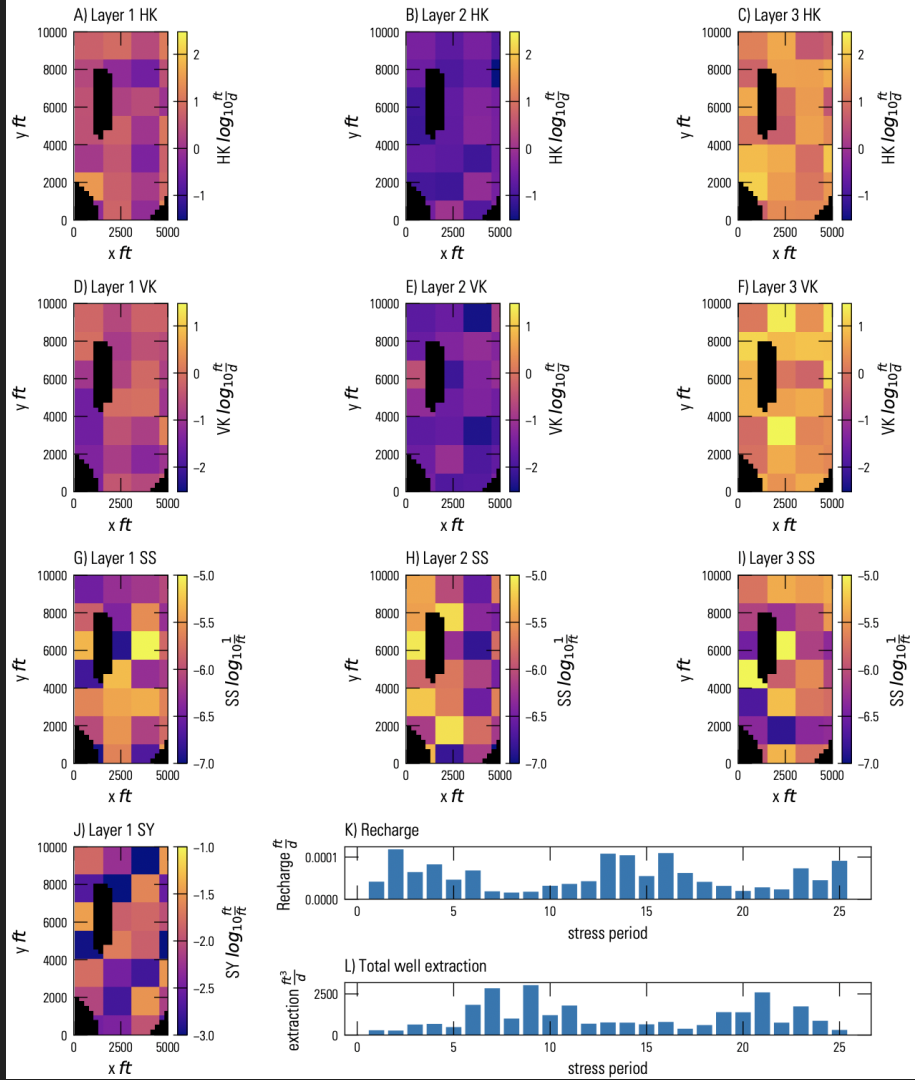
- Red = observed
- Grey = prior
- Blue = posterior



- “Best fit” parameter vector
- Blocky parameterization
- “Minimum error variance” solution
  - Previous GMDSI webinars
- Regularized = little variability

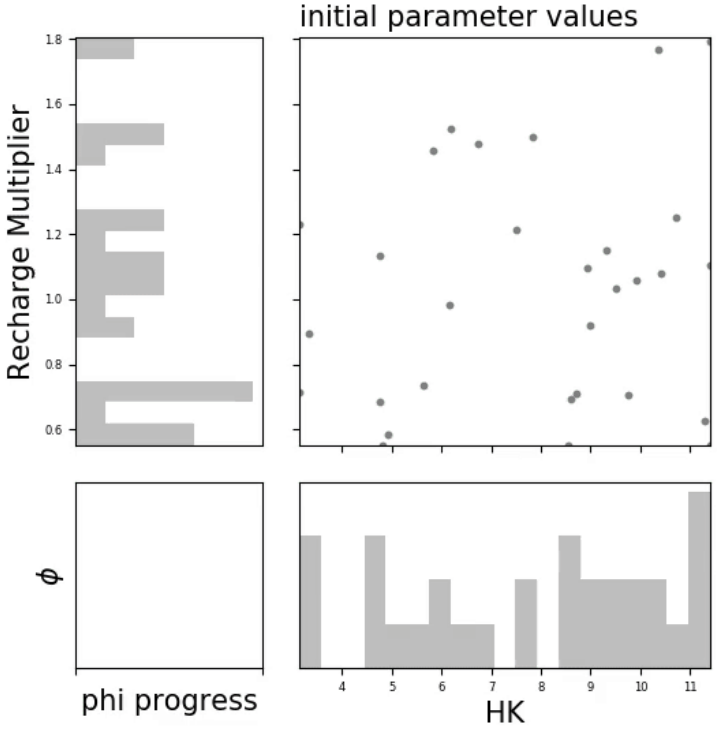


- A posterior realization
  - Drawn from the bayes-linear posterior covariance matrix
- Highly variable
- Consistent with Prior
- Reproduces historic observations



# PESTPP-IES: overview

- See previous GMDSI webinar on ensemble methods
- Iterative, localized ensemble smoother
- Focus on uncertainty
  - “ensemble”
- Moving on....

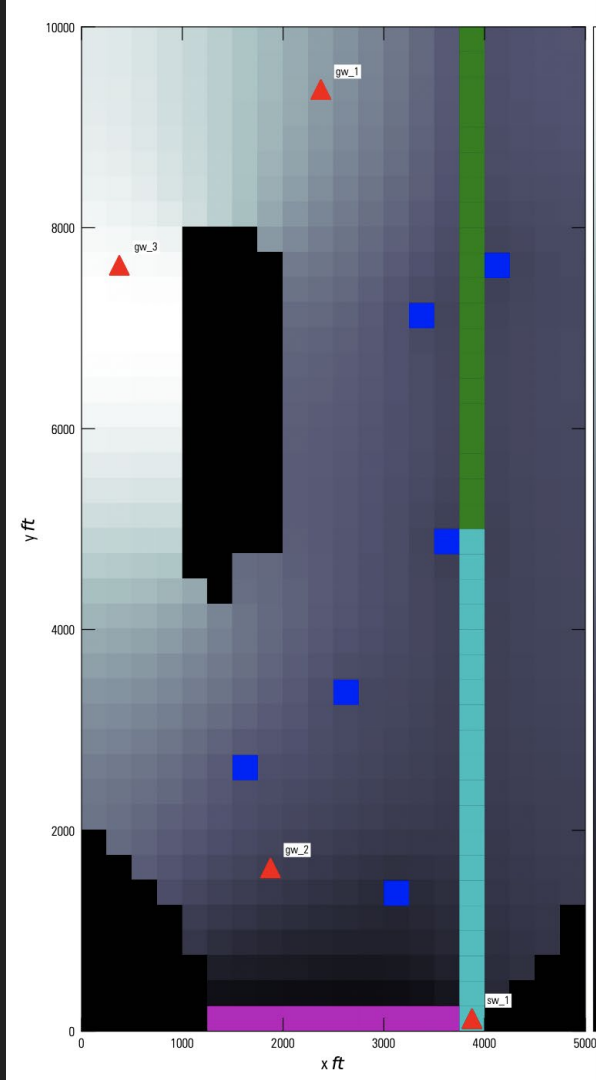



$$\underbrace{\mathbf{P}_{\Delta}}_{\text{upgrade matrix}} = -\Big(\underbrace{(\mathbf{J}^T\boldsymbol{\Sigma}_{\epsilon}^{-1}\mathbf{J})}_{\text{approx Hessian}} + \underbrace{(1+\lambda)\boldsymbol{\Sigma}_{\mathbf{P}}^{-1}}_{\text{dampening}}\Big)^{-1}\Big(\boldsymbol{\Sigma}_{\mathbf{P}}^{-1}\underbrace{(\mathbf{P}-\mathbf{P}_0)}_{\text{par change matrix}} + \mathbf{J}^T\boldsymbol{\Sigma}_{\epsilon}^{-1}\underbrace{(\mathbf{D}_{\text{sim}}-\mathbf{D}_{\text{obs}})}_{\text{innovations matrix}}\Big).$$

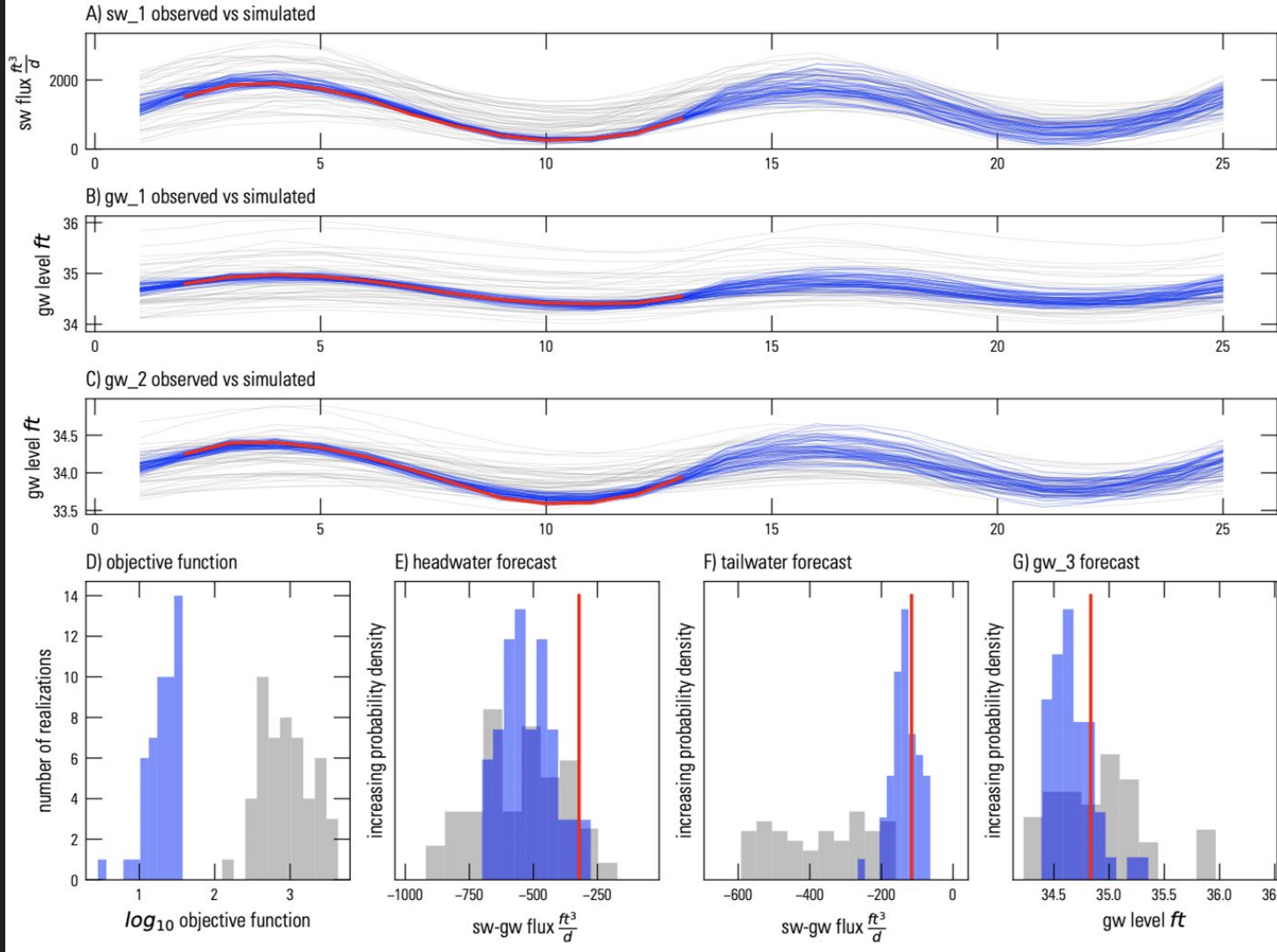
White (2018). A model-independent iterative ensemble smoother for history matching and uncertainty quantification in very high dimensions.

# PSETPP-IES: setup

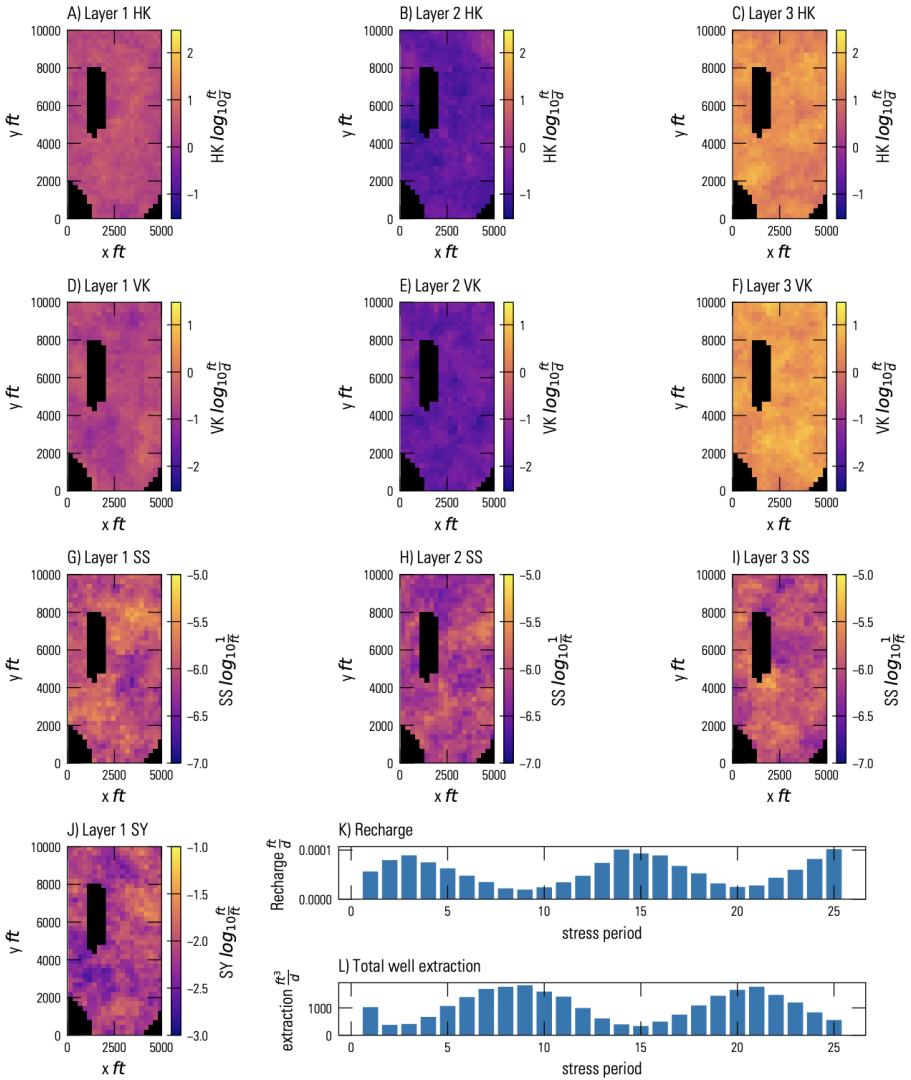
- Using the full complement of 8,175 parameters
- ++ies_parameter_ensemble(ies_prior_en.jcb)
  - pyEMU/PEST utilities
- ++ies_localizer(temporal_loc.jcb)
  - Eliminate spurious backwards-in-time correlations
- ++ies_autoadaloc(true)
- 378 total model runs



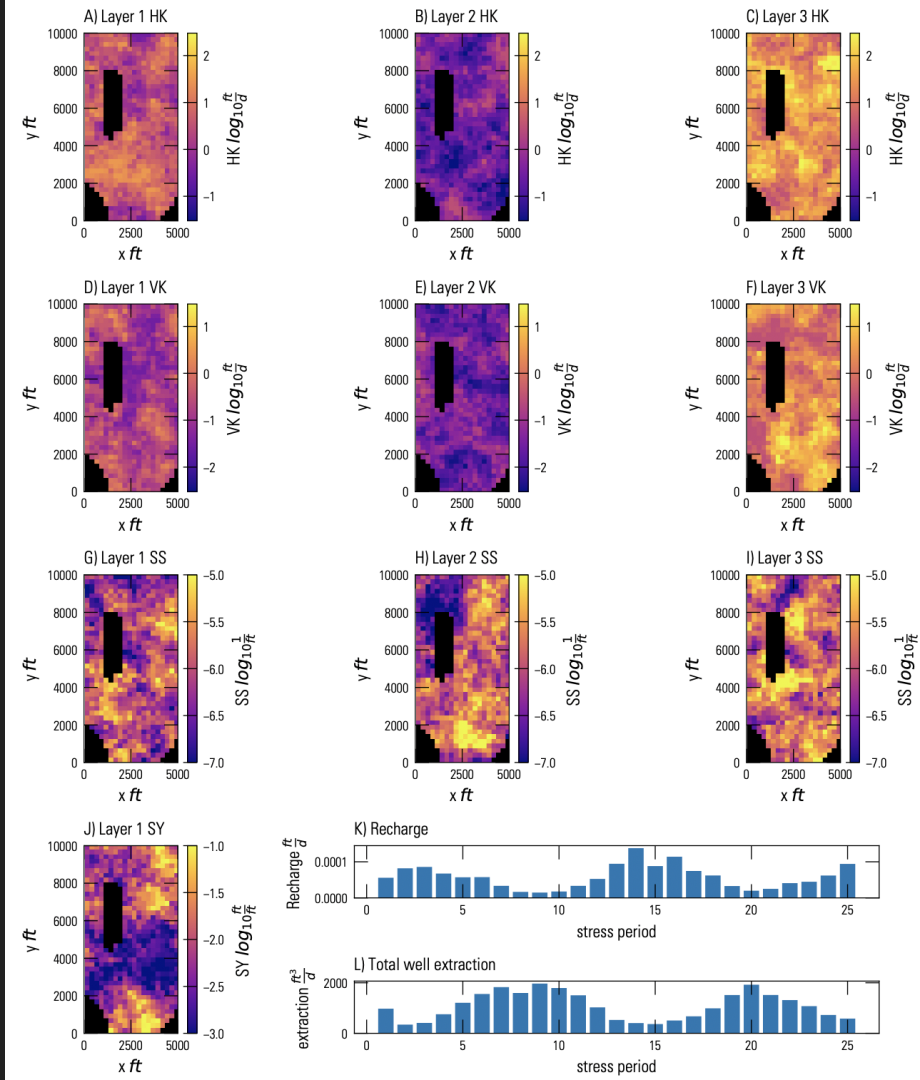
- Red = observed
- Grey = prior
- Blue = posterior



- “Base” parameter realization
- “Minimum error variance”-ish
- Regularized-ish
  - Reasonable variability
- Consistent with the Prior



- A posterior realization
- Highly variable
- Consistent with Prior
- Reproduces historic observations



# Now What?

- Run some scenarios and monitor “outputs of interest” (e.g. forecasts)
- Hope results are acceptable
  - Modify as needed
- Finish line

# Now What?

- Run some scenarios and monitor “outputs of interest” (e.g. forecasts)
- Hope results are acceptable
  - Modify as needed
- Finish line

**Critical thinking:**

“Acceptable” = Simulated forecast value is acceptable to decision makers

“Acceptable” = avoid bad outcomes

“Acceptable” = feasible solution

“Modify as needed” = seeking feasible solutions

# Mgmt opt under uncertainty

- Acceptable scenarios are feasible solutions
- Usually suboptimal
  - there is a “better” (optimal) way to manage
- mgmt optimization formalizes the scenario-testing process
  - Make/save $ while ensuring bad things don’t happen
- Can frame results in **$** and **risk** (uncertainty continued…)

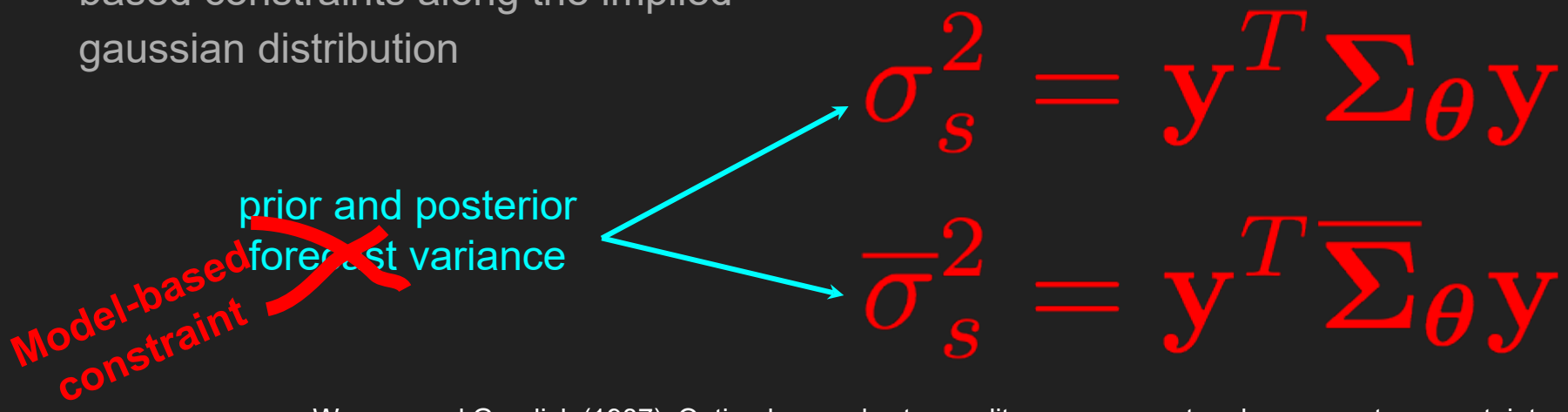
# UQ/DA: terminology vs Mgmt Optimization

- Parameters
- (historic) observations
- **forecast/prediction** → **Model-based constraints**
- Objective function
- Prior information
- Jacobian matrix

- Decision variables
- ?
- **Model-based constraints**
- Objective function
- Prior information constraints
- Response matrix

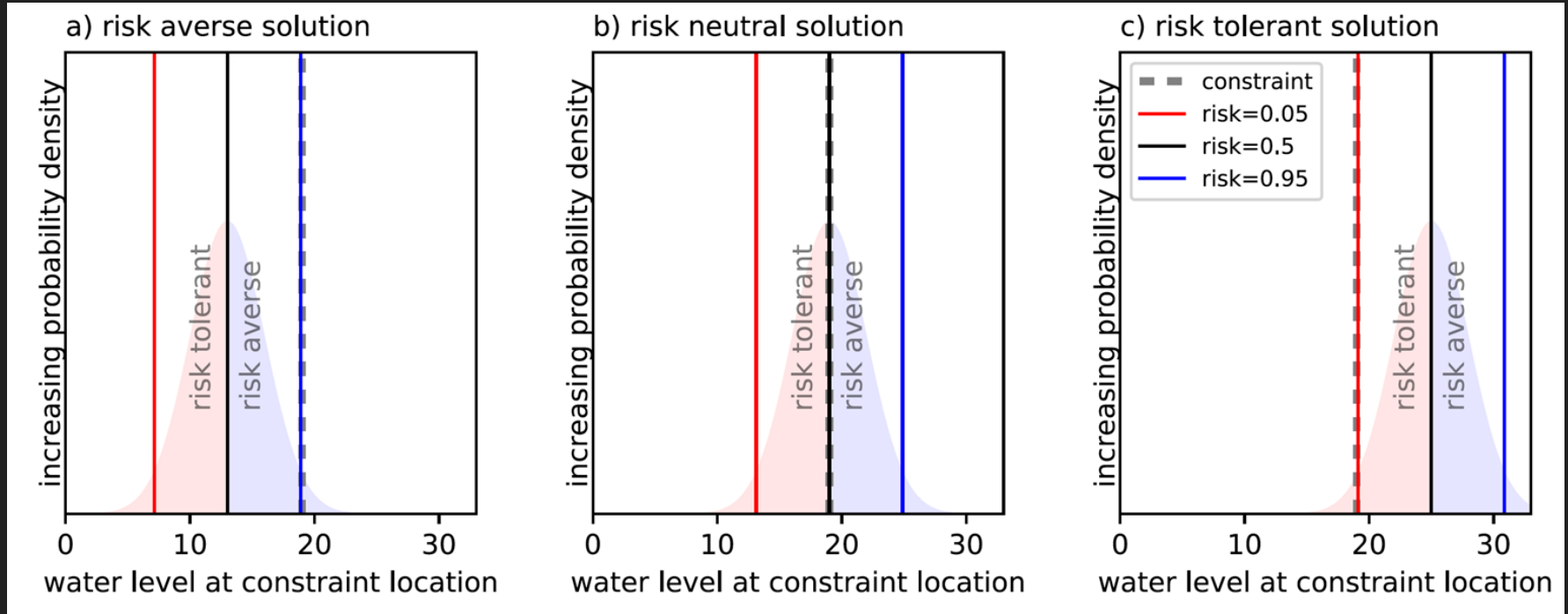
# Chance constraints: Theory

- We can use the same FOSM trickery to estimate uncertainty in model-based constraints
- concept of “risk” to “shift” the model-based constraints along the implied gaussian distribution



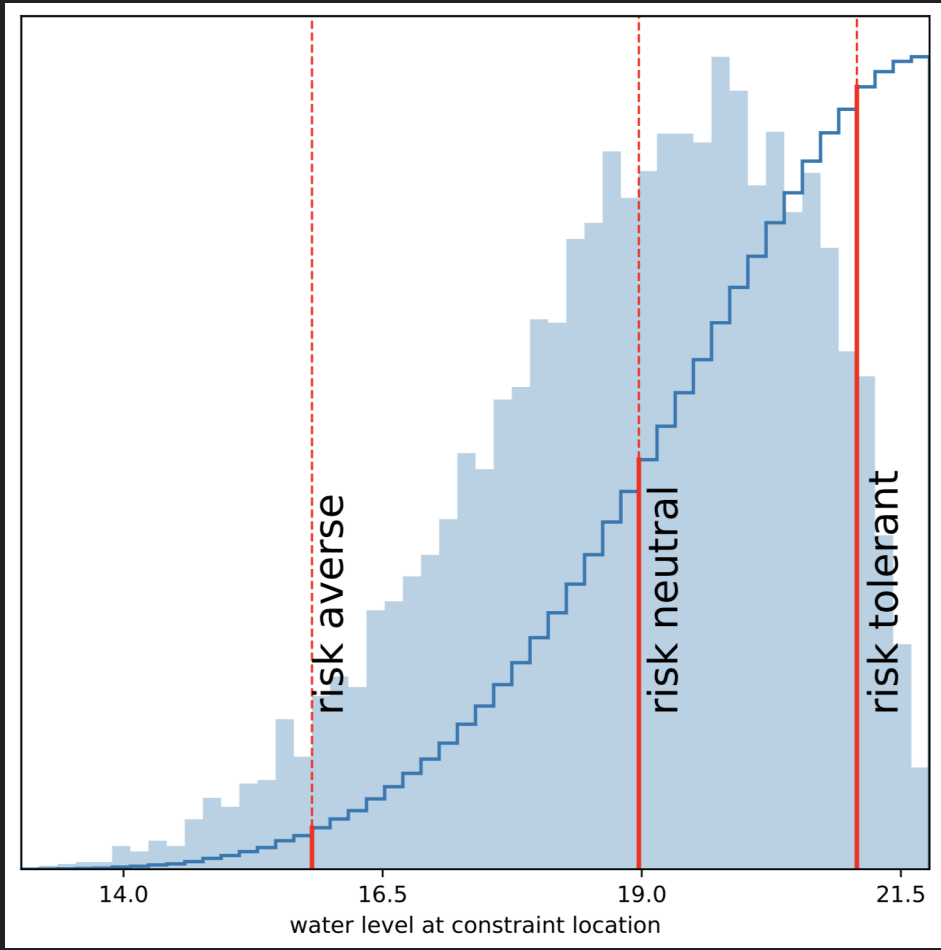

Wagner and Gorelick (1987). Optimal groundwater quality management under parameter uncertainty.

# Chance constraints: Theory



# Chance constraints: Theory

- Use ensembles to represent model-based constraint uncertainty
- “Stack-based” optimization
- Less assumptions
  - “non-parametric”




Bayer, and others. 2010. Optimization of high-reliability hydrological design problems by robust automatic sampling of critical model realizations.

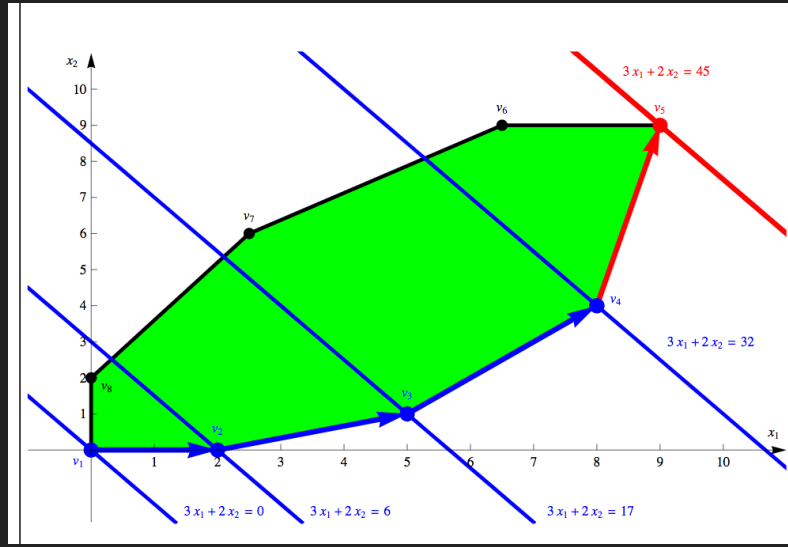
# PESTPP-OPT: overview

- Sequential linear programming with chance constraints
- “Mildly” nonlinear relations
- Parallel run mgmt for filling response matrix
- FOSM and stack chance constraints

$$\text{minimize:} \mathbf{c}^{\mathbf{T}}\mathbf{x}$$

$$\text{subject to: } \mathbf{Ax} \leq \mathbf{b}$$

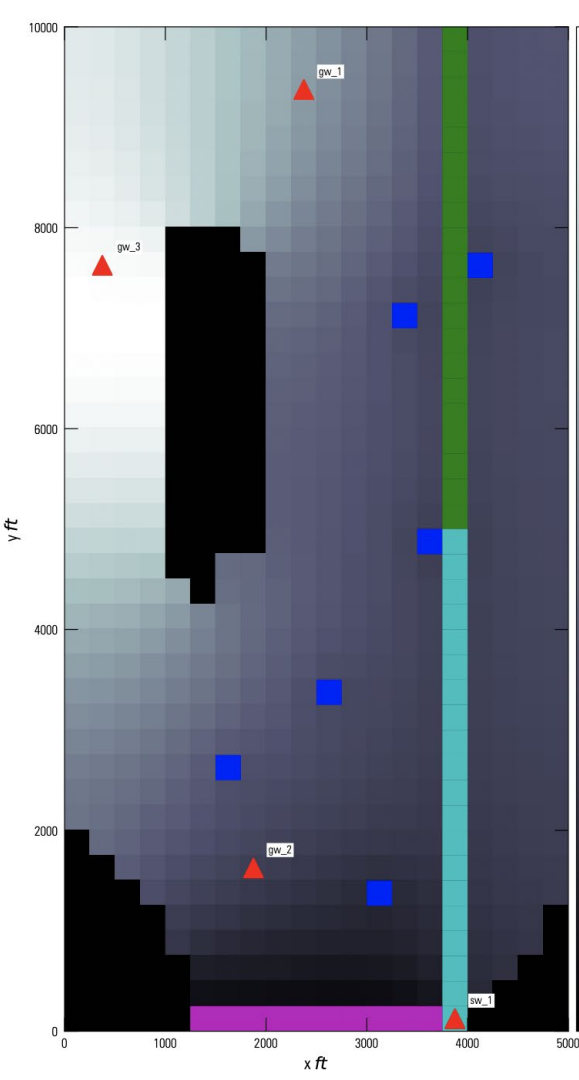
$$\mathbf{x} \geq \mathbf{0}$$




White and others (2018). A tool for efficient, model-independent management optimization under uncertainty

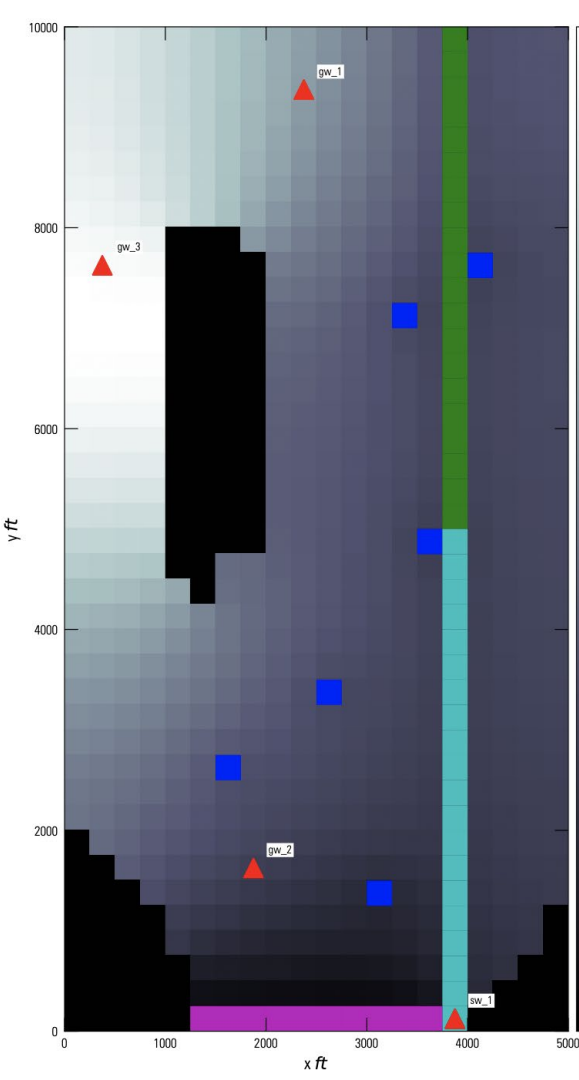
# PESTPP-OPT: setup

- setup wel flux parameters as decision variables
  - 6 wels X 24 stress periods = 144 wel flux decision variables
- add prior information equations for minimum required wel flux
  - At least 750 cfd
  - Maintain water supply (demand-side)
- Identify simulated sw-gw exchange as constraints
  - Maintain eco flows (supply-side)
  - At least 250 cfd for headwater and tailwater
  - Uncertain! So chance constraints



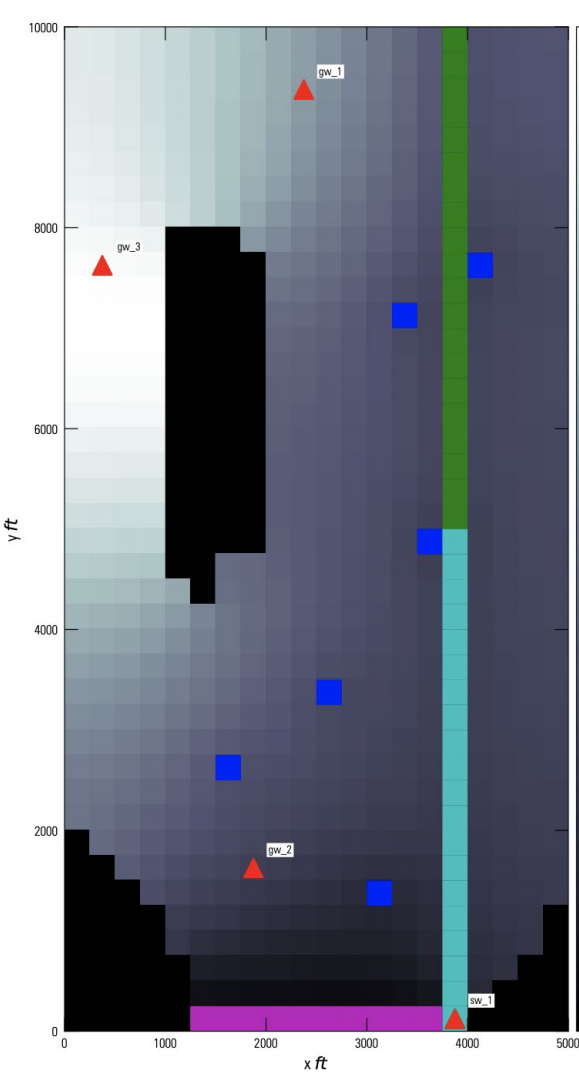
# PESTPP-OPT: setup

- 3 PESTPP-OPT runs
- Risk neutral
- Risk-averse with FOSM-based chance constraints
  - Using only 25 stress-period recharge parameters
- Risk-averse with stack-based chance constraints
  - posterior PESTPP-IES parameter ensemble

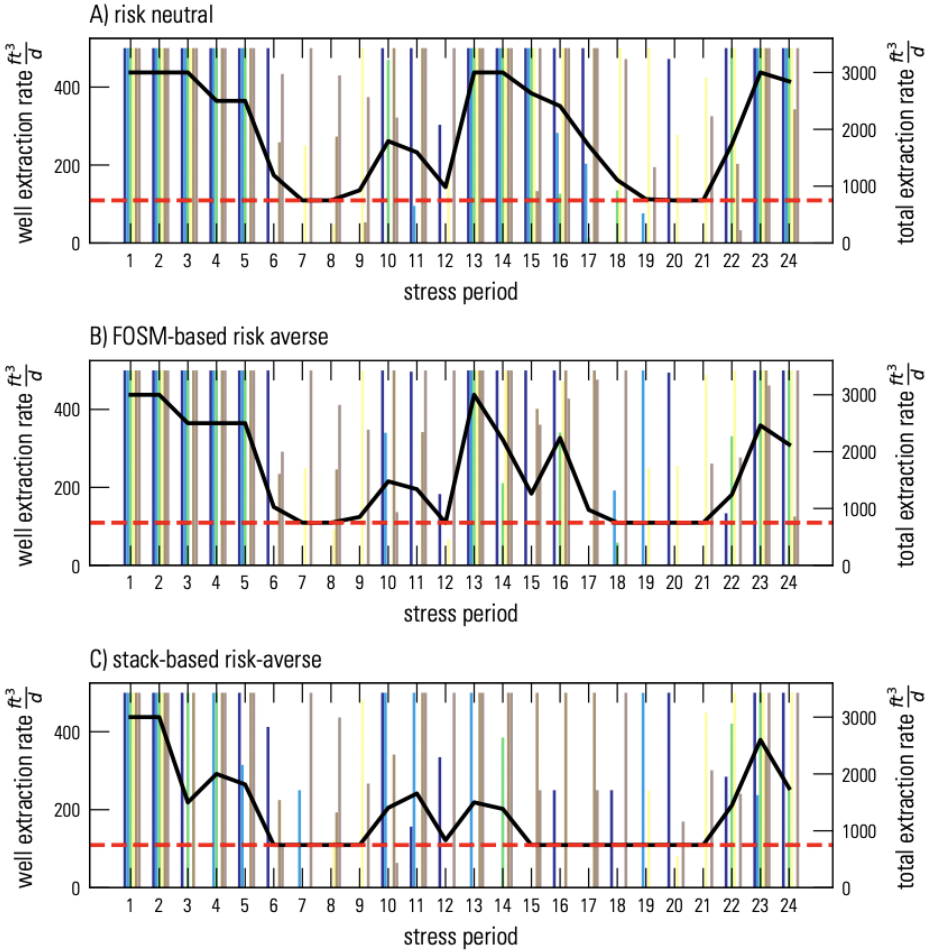


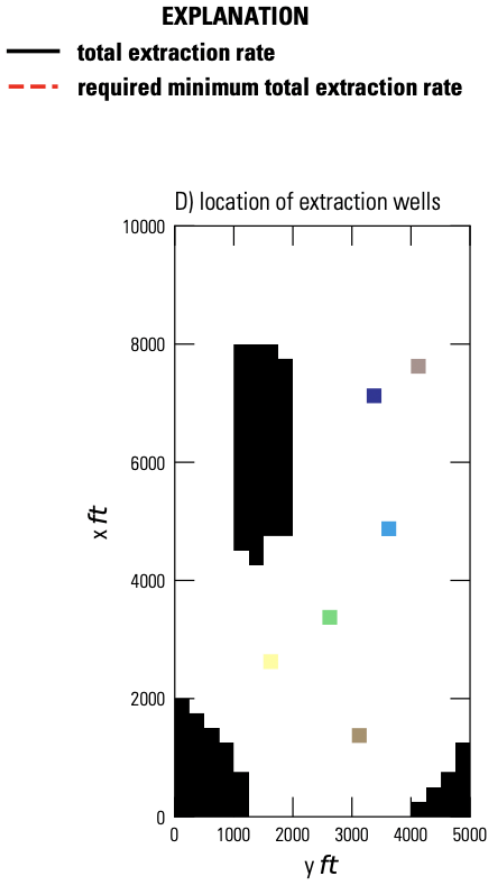
# PESTPP-OPT: setup

- ++opt_direction(max)
- ++opt_risk(0.95)
- ++opt_par_stack(par_stack.csv)
- 147 total model runs for risk neutral
- 172 total model runs for FOSM-based risk averse
- 197 total model runs for stack-based risk averse

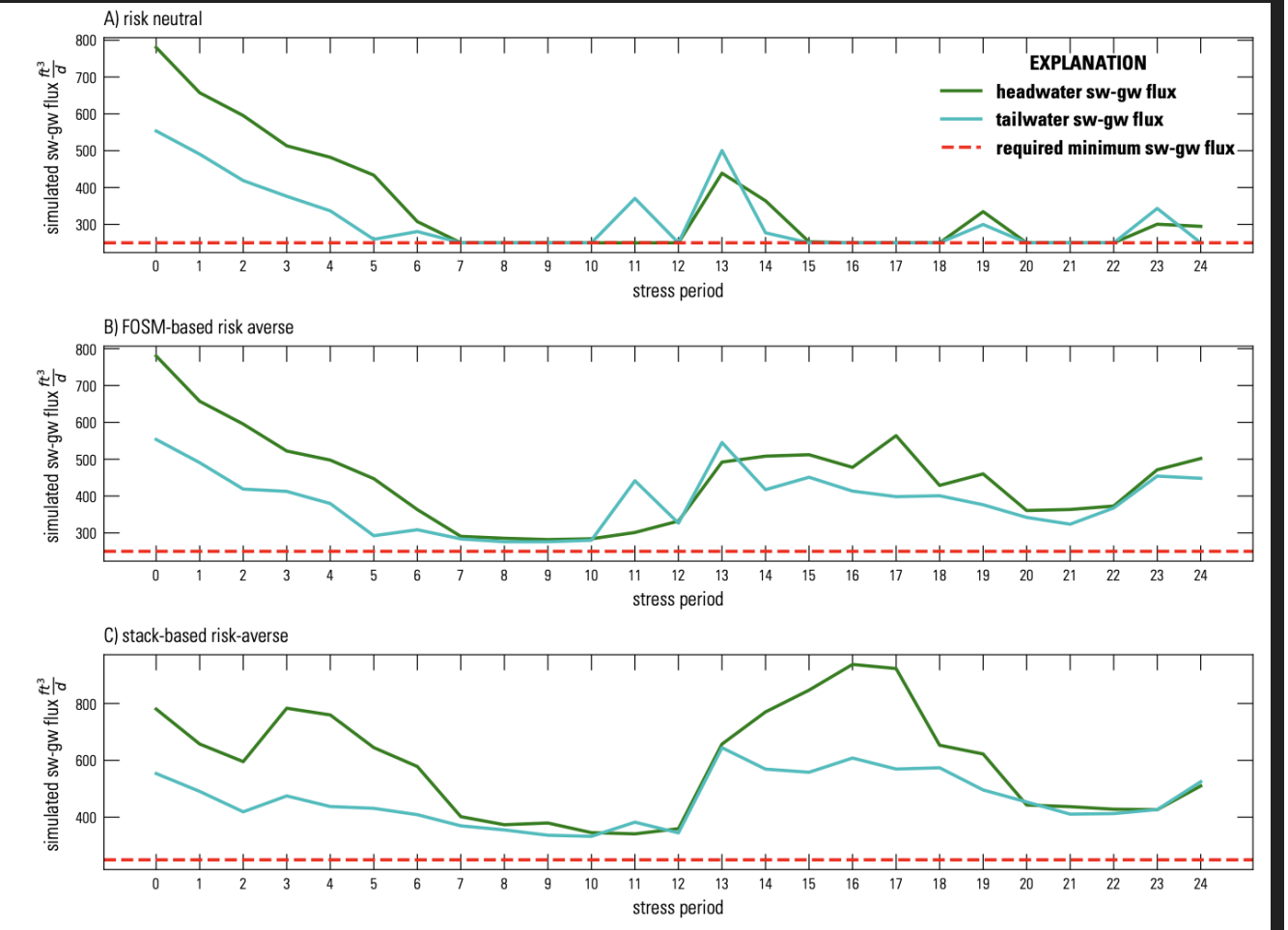


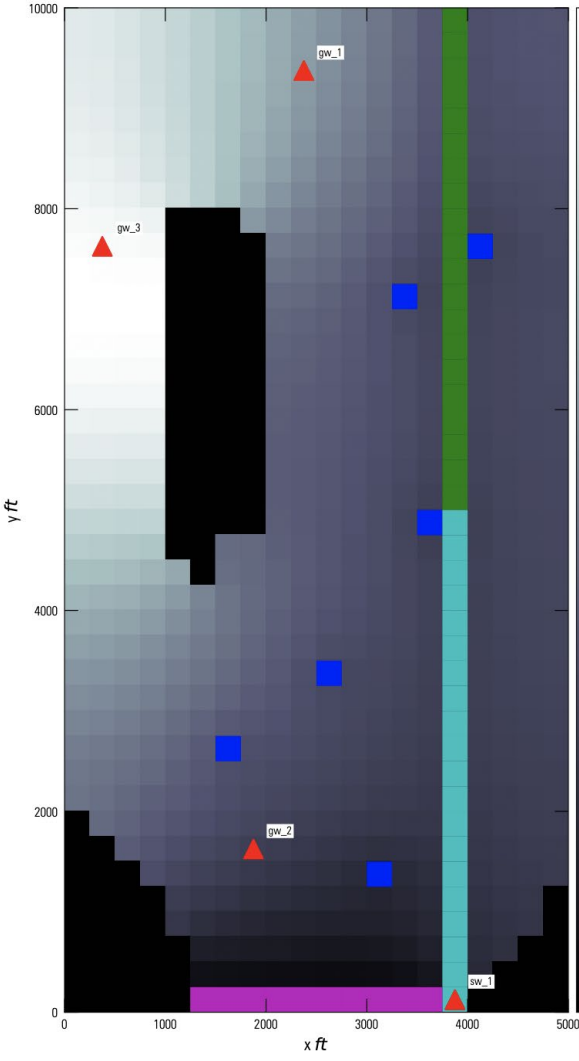
# PESTPP-OPT: results



# PESTPP-OPT: results



# Python-full Example Workflow

https://github.com/pypest/pyemu_pestpp_workflow

- Currently includes jupyter notebooks demonstrating
  - Programmatically setup a single “comprehensive” PEST++ interface (pyEMU PstFrom)
  - Running and plotting Prior Monte Carlo
- More notebooks are being added...

# Python-full Example Workflow

PstFrom: setup

```
pf = PstFrom(original_d=tmp_model_ws, new_d=template_ws,
                    remove_existing=True,
                    longnames=True, spatial_reference=sr,
                    zero_based=False, start_datetime="1-1-2018")
```

# Python-full Example Workflow

PstFrom: setup

```
pf.add_observations("sfr.csv", insfile="sfr.csv.ins", index_cols="time",
                    use_cols=["gage_1", "headwater", "tailwater"],
                    ofile_sep=",")
```

# Python-full Example Workflow

PstFrom: setup

```
rch_files = ["recharge_1.txt","recharge_2.txt","recharge_3.txt"]

pf.add_parameters(filenames=rch_files, par_type="pilotpoints",par_name_base="rch_pp",

                  pargp="rch_pp", zone_array=ib, upper_bound=0.8, lower_bound=1.2,

                  geostruct=rch_gs)
```

# Python-full Example Workflow

PstFrom: setup

```
pf.add_parameters(filenames=”hk_layer_1.txt”, par_type="grid",
                  par_name_base="hk1_gr", pargp="hk1_grid", zone_array=ib,
                  upper_bound=0.5, lower_bound=50.0, geostruct=rch_gs,
                  par_style=”direct")
```

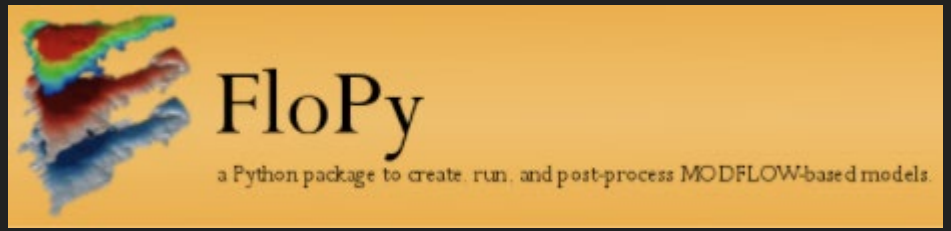
# Python-full Example Workflow

PstFrom: setup

```
pf.mod_sys_cmds.append("mf6")

pst = pf.build_pst('freyberg.pst')

pe = pf.draw(num_reals, use_specsim=True)

pe.to_csv("prior.csv")

cov = pf.buid_prior()

cov.to_uncfile("prior.unc")
```

# A higher resolution (synthetic) model

If you happen to be using MODFLOW, flopy is you tool!



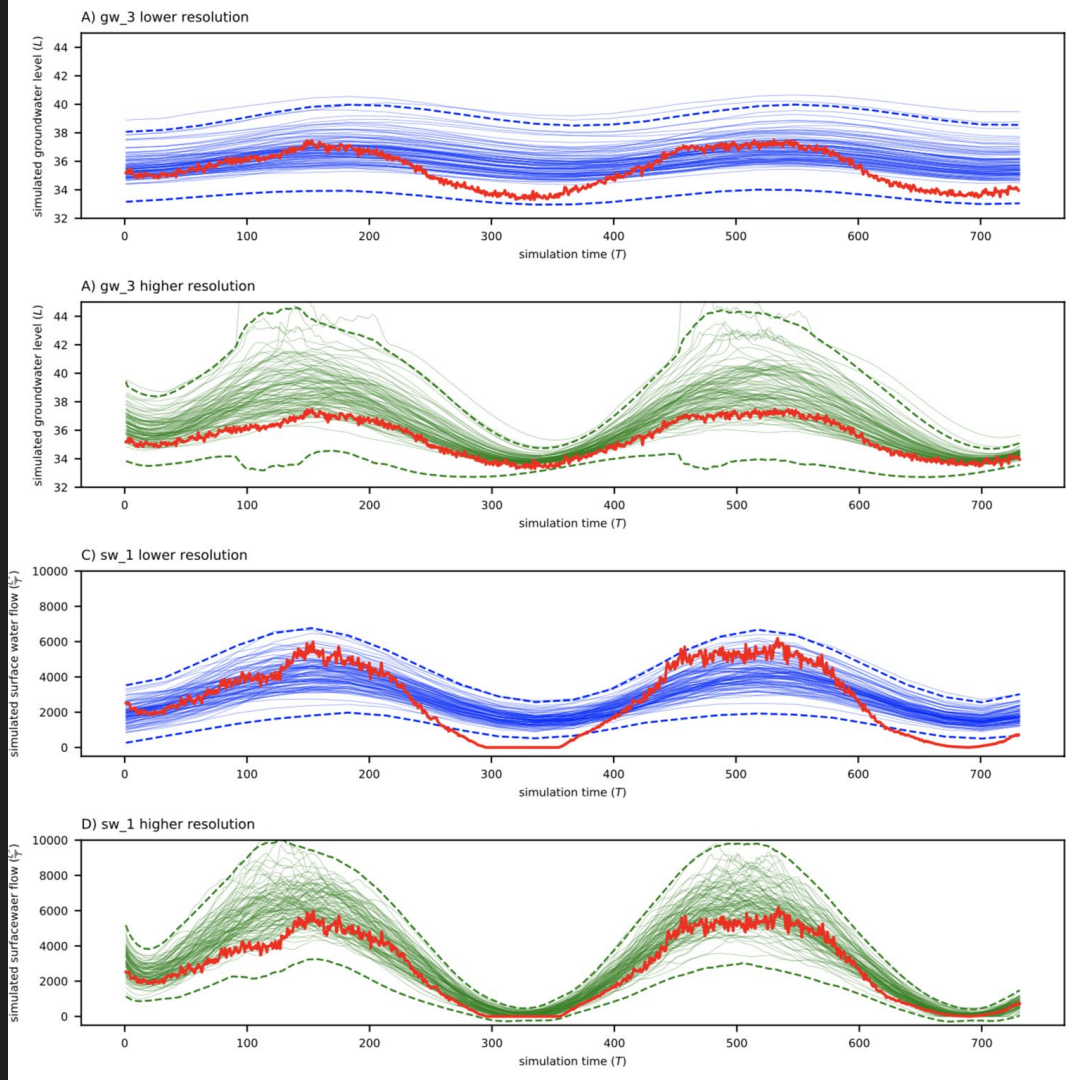

...but the same PstFrom script

- 3 layers X 120 rows X 60 columns
- 730 daily stress periods

Imagine how disruptive changing discretization is….

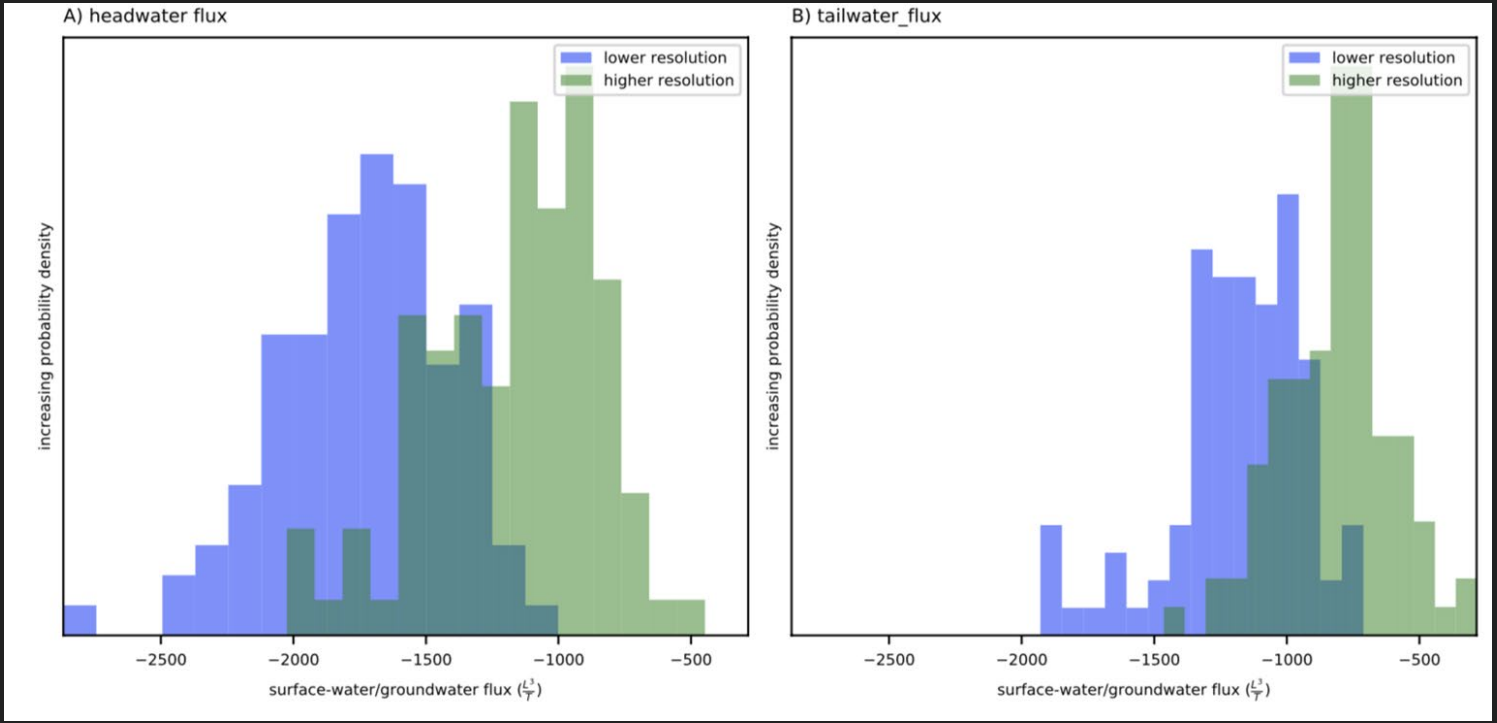
# Python-full Example Workflow

Prior Monte Carlo results



# Python-full
# Example Workflow

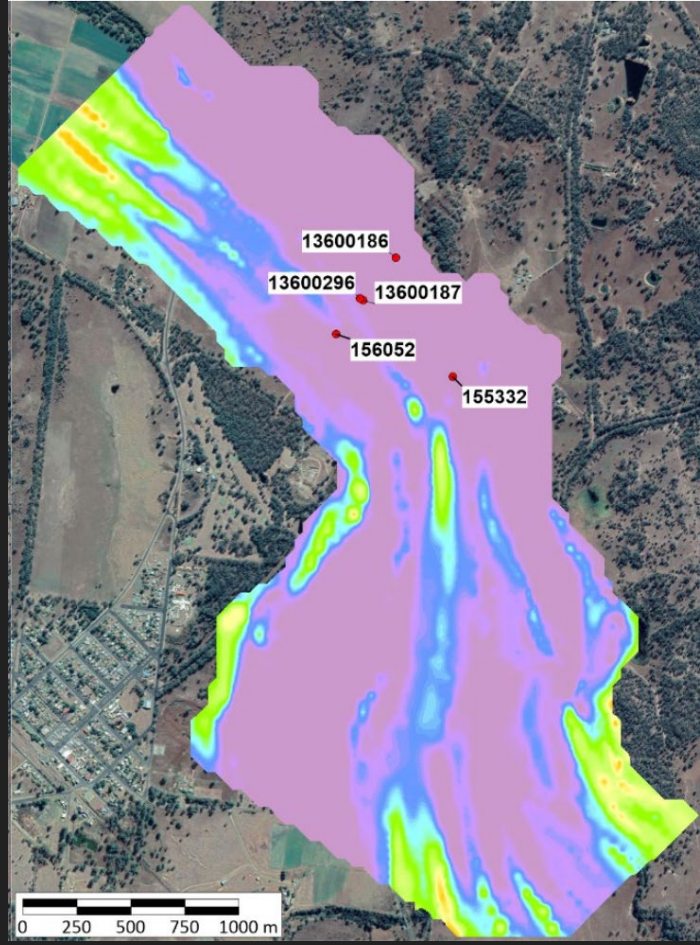
Prior Monte Carlo results



# Looking forward

PEST++

- Off loading linear algebra to workers
  - Needed for very high dimensional localization solve
- Geostats++
  - Tighter integration of estimation and simulation
  - Unstructured grids
  - Conditional simulation
  - Also for pyEMU

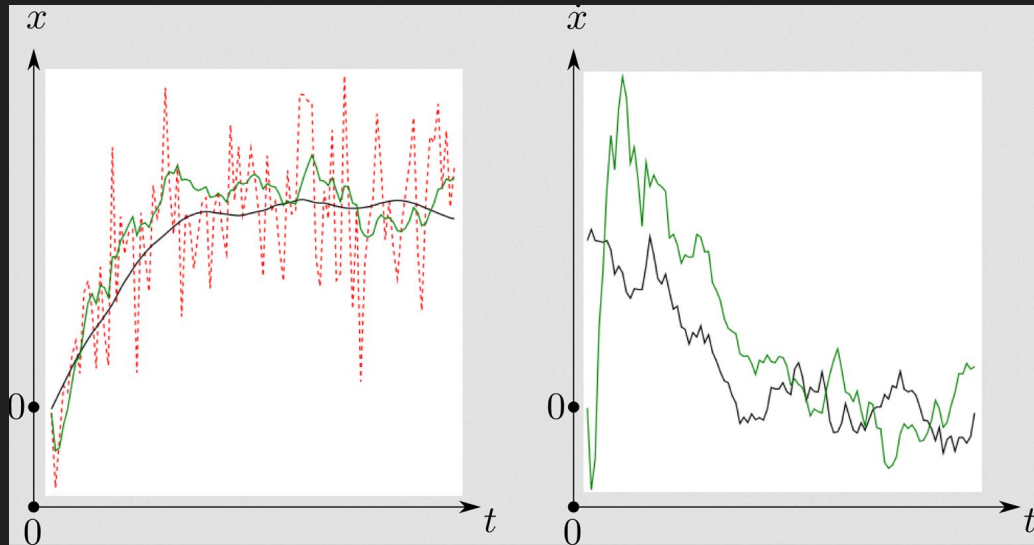



Gallagher and Doherty. 2020. Water supply security for the township of Biggenden. A GMDSI worked example report

# Looking forward

## PESTPP-DA

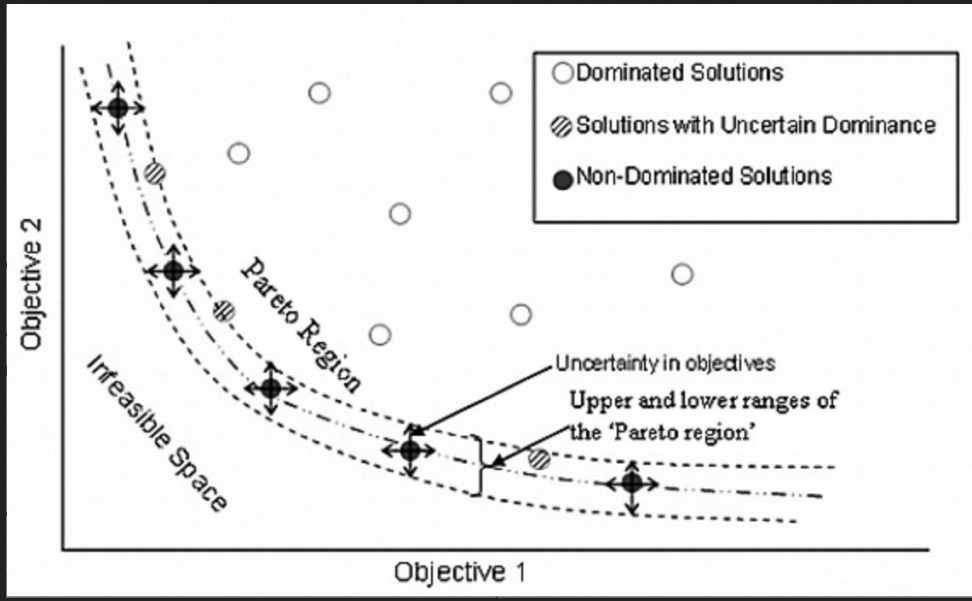
- The ensemble Kalman suite
  - **Iterative ensemble Kalman Filter**
  - Multiple Data Assimilation approach
- sequential/recursive estimation
- Forward-in-time processes
  - Subsidence
  - Mass transport
  - Saltwater intrusion
- Ayman Alzraiee (USGS)



# Looking forward

## PESTPP-MOU

- Constrained multi-objective optimization
  - evolutionary/genetic global algorithms
- Chance constraints
  - FOSM and stacks
- Zak Stanko (USGS)




Singh and Minsker 2008. Uncertainty-based multiobjective optimization of groundwater remediation design

# Looking forward

## PESTPP-SQP

- Industrial-strength constrained non-linear optimization
  - Ensemble approximation
  - Constrained quasi-newton solution
- Chance constraints
  - FOSM and stacks
- Matt Knowling (Uni Adelaide)

# Looking forward

## pyEMU and PstFrom

- Higher-level interface(s)
  - Spreadsheets
  - yml/json
  - GUI
- Better support for unstructured grids
- Improved geostats
  - Improved speed
  - Conditional sim

# Where are these tools and examples?

- https://github.com/usgs/pestpp
- https://github.com/pypest/pyemu
- https://github.com/pypest/pyemu_pestpp_workflow
- PEST++ Version 5 USGS report

And for python with MODFLOW

- https://github.com/modflowpy/flopy
- https://github.com/aleaf/modflow-setup

**Thanks for listening!**