How to tune your simulator

Jonathan Rougier

Rougier Consulting Ltd
& University of Bristol

Barnett Lecture
RSS Annual Conference, Sep 2021

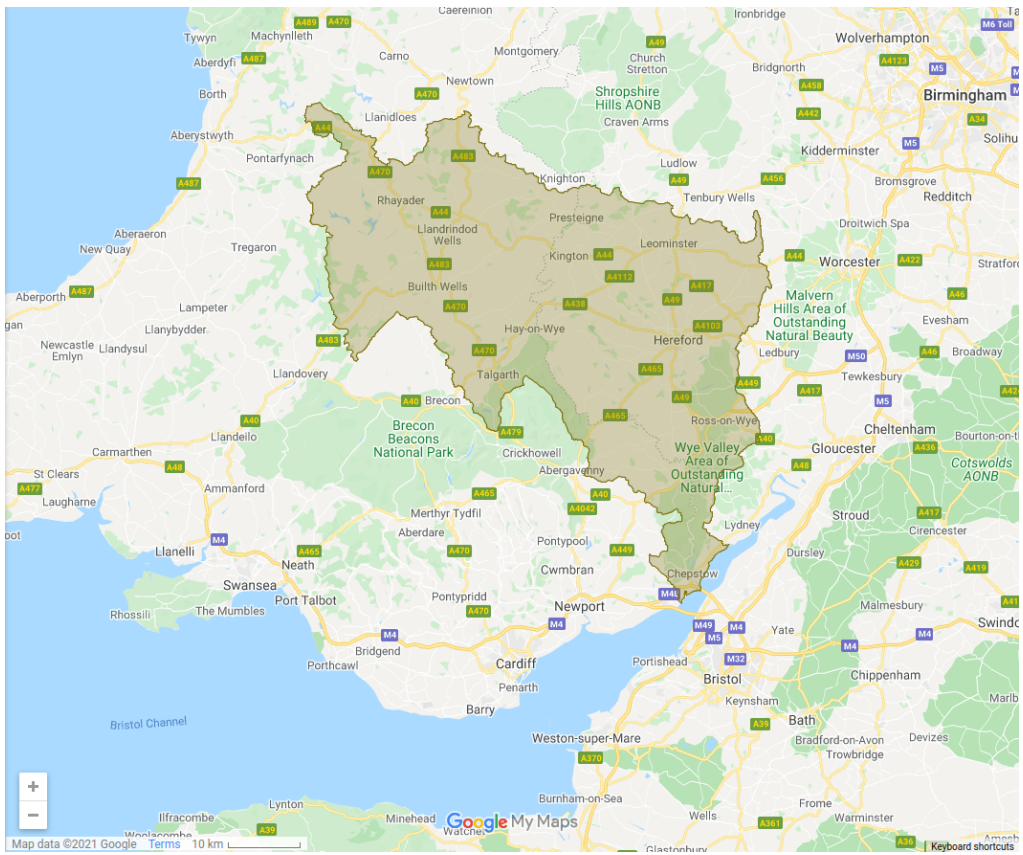
Overview

There are lots of applications where a computer simulator is used to map a time-series of inputs (forcing) into a time-series of outputs.

Overview

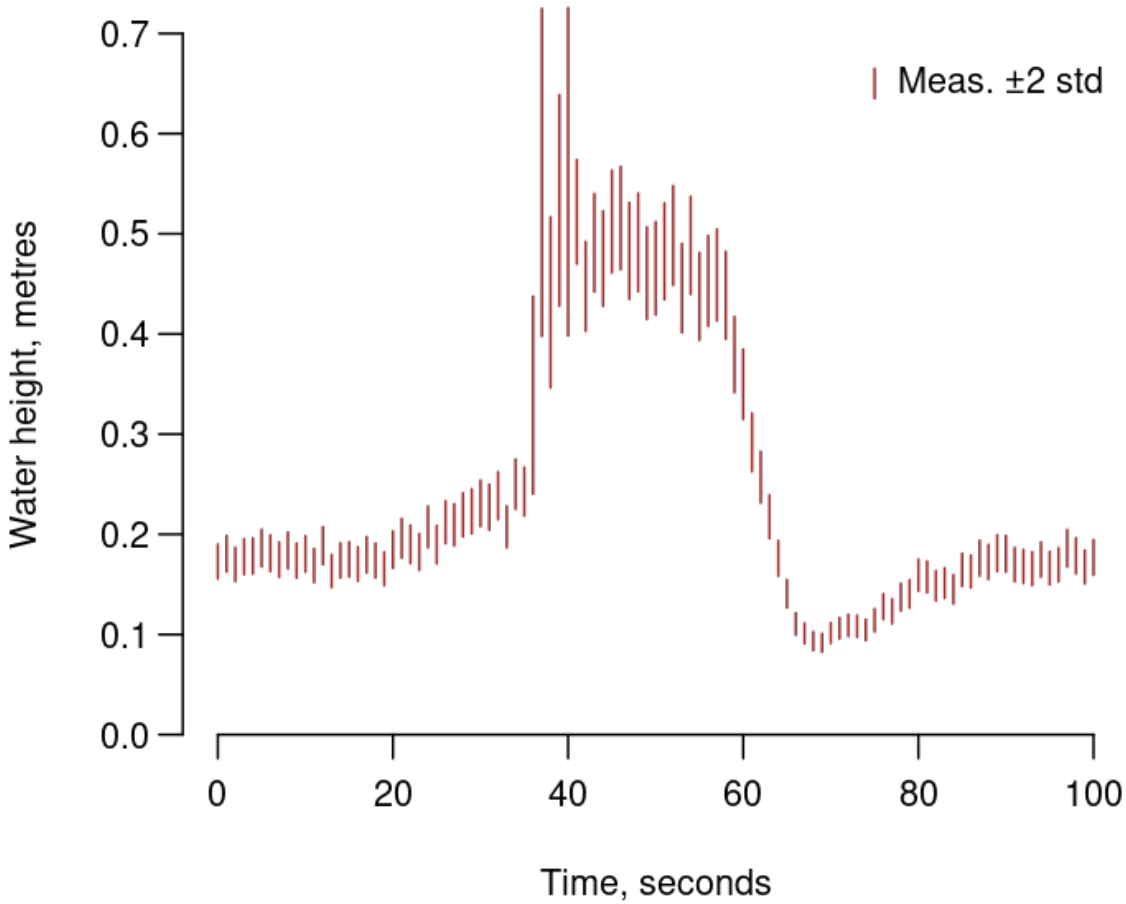
There are lots of applications where a computer simulator is used to map a time-series of inputs (forcing) into a time-series of outputs. E.g., **flood modelling in Environmental Science:**

The Wye catchment and Hereford



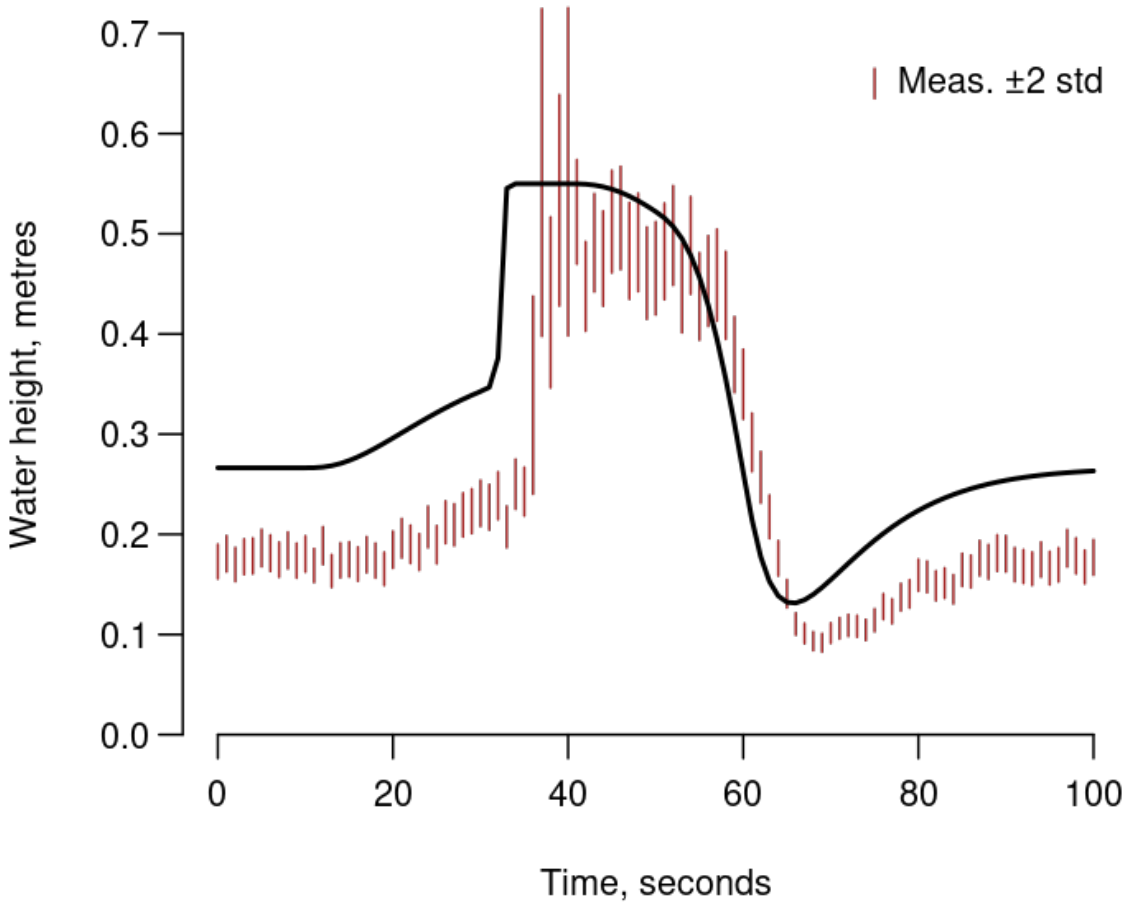
Outline of our application

Nature gives us this 😕



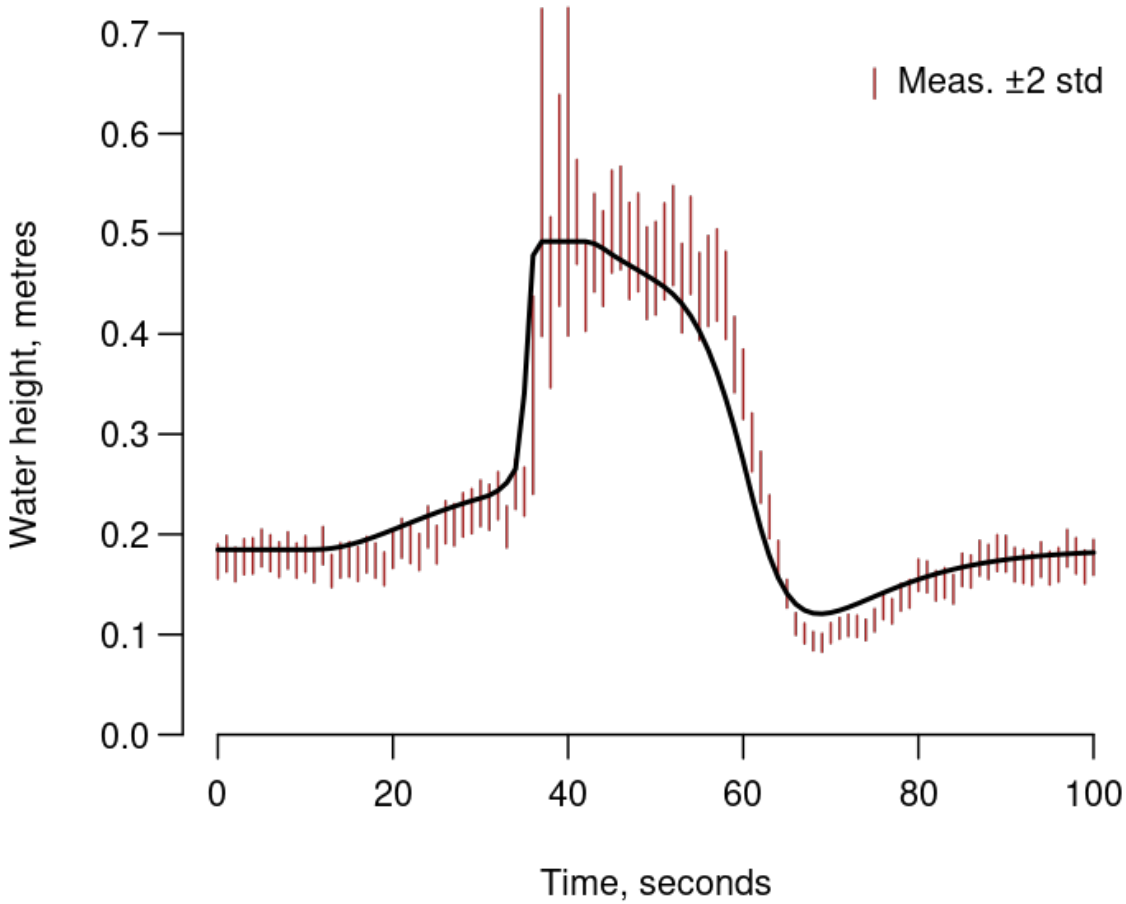
Outline of our application

Our first run of the simulator gives us this 😱



Outline of our application

So how do we efficiently get to this? 😌



Outline of my approach

- There is an ‘engineering’ approach which says minimize the misfit between the simulator output and the observations by varying the simulator parameters, where ‘misfit’ is usually sum of squared errors.

Outline of my approach

- There is an 'engineering' approach which says minimize the misfit between the simulator output and the observations by varying the simulator parameters, where 'misfit' is usually sum of squared errors.
- I will show how applying a more statistical approach can lead to a better outcome, even under the 'engineering' criterion.

Outline of my approach

- There is an 'engineering' approach which says minimize the misfit between the simulator output and the observations by varying the simulator parameters, where 'misfit' is usually sum of squared errors.
- I will show how applying a more statistical approach can lead to a better outcome, even under the 'engineering' criterion.

A. In a statistical approach we link the simulator parameters and the observations in a statistical model, which explicitly allows for limitations in the simulator.

Outline of my approach

- There is an 'engineering' approach which says minimize the misfit between the simulator output and the observations by varying the simulator parameters, where 'misfit' is usually sum of squared errors.
- I will show how applying a more statistical approach can lead to a better outcome, even under the 'engineering' criterion.

A. In a statistical approach we link the simulator parameters and the observations in a statistical model, which explicitly allows for limitations in the simulator.

B. This produces a smoother objective function, and then we can use statistical optimization to manage the trade-off between 'explore' and 'exploit' with only a limited number of simulator runs.

A little notation

- f is the simulator, with a scalar output.
- Its inputs comprise control variables x and parameters θ.
- Collectively, $\boldsymbol{x} := (x_1, \ldots, x_n)$ and

$$f(\boldsymbol{x}; \theta) := \begin{pmatrix} f(x_1; \theta) \\ \vdots \\ f(x_n; \theta) \end{pmatrix} = \begin{pmatrix} f_1(\theta) \\ \vdots \\ f_n(\theta) \end{pmatrix} = \boldsymbol{f}(\theta).$$

- Similarly,

$$Y(\boldsymbol{x}) := \begin{pmatrix} Y_1 \\ \vdots \\ Y_n \end{pmatrix} = \boldsymbol{Y}, \quad Z(\boldsymbol{x}) := \begin{pmatrix} Z_1 \\ \vdots \\ Z_n \end{pmatrix} = \boldsymbol{Z}$$

are actual system values, and observables, respectively.

The Normal model

- Assume a transformation of $\boldsymbol{Z}$, $\boldsymbol{Y}$, and $\boldsymbol{f}$ after which

$$\boldsymbol{Z} \mid \boldsymbol{Y}, \theta^*, \alpha, K \sim \mathrm{N}\left(\boldsymbol{Y}, D\right)$$
$$\boldsymbol{Y} \mid \theta^*, \alpha, K \sim \mathrm{N}\left(\alpha \mathbf{1} + \boldsymbol{f}(\theta^*), K\right)$$

is an acceptable implementation of the ▸ 'best input' model.

  - $D := \mathrm{diag}(\sigma_1^2, \ldots, \sigma_n^2)$, a diagonal matrix of reported measurement errors,
  - θ^* is the best value of the parameters,
  - α is the scalar offset,
  - K is the $n \times n$ discrepancy variance.

The Normal model

- Assume a transformation of $\boldsymbol{Z}$, $\boldsymbol{Y}$, and $\boldsymbol{f}$ after which

$$\boldsymbol{Z} \mid \boldsymbol{Y}, \theta^*, \alpha, K \sim \mathrm{N}\left(\boldsymbol{Y}, D\right)$$
$$\boldsymbol{Y} \mid \theta^*, \alpha, K \sim \mathrm{N}\left(\alpha \mathbf{1} + \boldsymbol{f}(\theta^*), K\right)$$

  is an acceptable implementation of the ▸ 'best input' model.

  - $D := \mathrm{diag}(\sigma_1^2, \ldots, \sigma_n^2)$, a diagonal matrix of reported measurement errors,
  - θ^* is the best value of the parameters,
  - α is the scalar offset,
  - K is the $n \times n$ discrepancy variance.

- Integrating out $\boldsymbol{Y}$,

$$\boldsymbol{Z} \mid \theta^*, \alpha, K \sim \mathrm{N}\Big(\alpha \mathbf{1} + \boldsymbol{f}(\theta^*), K + D\Big),$$

  from which derive $L(\theta^*, \alpha, K)$, based on measurements $\boldsymbol{z}^{\mathrm{obs}}$.

The Normal model (cont)

- We can use the deviance,

$$-2 \log L(\theta^*, \alpha, K),$$

as a measure of misfit. There are a couple of nuisance parameters 🤔

The Normal model (cont)

- We can use the deviance,

$$-2 \log L(\theta^*, \alpha, K),$$

as a measure of misfit. There are a couple of nuisance parameters 🤔

- The 'engineering' approach eliminates the nuisance parameters by setting them to zero,

$$-2 \log L^{\text{eng}}(\theta^*) := -2 \log L(\theta^*, 0, \mathbf{0}) = \sum_{i=1}^{n} \frac{\left(z_i^{\text{obs}} - f_i(\theta^*)\right)^2}{\sigma_i^2},$$

apparently hoping that treating the simulator as perfect will make its limitations go away 😬

Can we do better?

We want to do something about those two nuisance parameters, other than setting them both to zero.

Can we do better?

We want to do something about those two nuisance parameters, other than setting them both to zero.

- Treat K as stationary, and thin z^{obs} to the point where $K_{\mathcal{I},\mathcal{I}} \approx \kappa^2 I$, where $\mathcal{I} \subset \{1, \ldots, n\}$ are the retained outputs:

$$-2 \log L_{\mathcal{I}}(\theta^*, \alpha, \kappa) \approx \sum_{i \in \mathcal{I}} \frac{\{z_i^{\text{obs}} - (\alpha + f_i(\theta^*))\}^2}{\kappa^2 + \sigma_i^2} + \sum_{i \in \mathcal{I}} \log(\kappa^2 + \sigma_i^2),$$

Can we do better?

We want to do something about those two nuisance parameters, other than setting them both to zero.

- Treat K as stationary, and thin $\boldsymbol{z}^{\text{obs}}$ to the point where $K_{\mathcal{I},\mathcal{I}} \approx \kappa^2 I$, where $\mathcal{I} \subset \{1, \ldots, n\}$ are the retained outputs:

$$-2 \log L_{\mathcal{I}}(\theta^*, \alpha, \kappa) \approx \sum_{i \in \mathcal{I}} \frac{\{z_i^{\text{obs}} - (\alpha + f_i(\theta^*))\}^2}{\kappa^2 + \sigma_i^2} + \sum_{i \in \mathcal{I}} \log(\kappa^2 + \sigma_i^2),$$

- Profile out α and κ (1D numerical optimization) to give thin & prof likelihood, ▸ More details

$$L_{\mathcal{I}}^{\text{prf}}(\theta^*) := \max_{\alpha,\kappa} L_{\mathcal{I}}(\theta^*, \alpha, \kappa)$$

Can we do better?

We want to do something about those two nuisance parameters, other than setting them both to zero.

- Treat K as stationary, and thin $\boldsymbol{z}^{\text{obs}}$ to the point where $K_{\mathcal{I},\mathcal{I}} \approx \kappa^2 I$, where $\mathcal{I} \subset \{1, \ldots, n\}$ are the retained outputs:

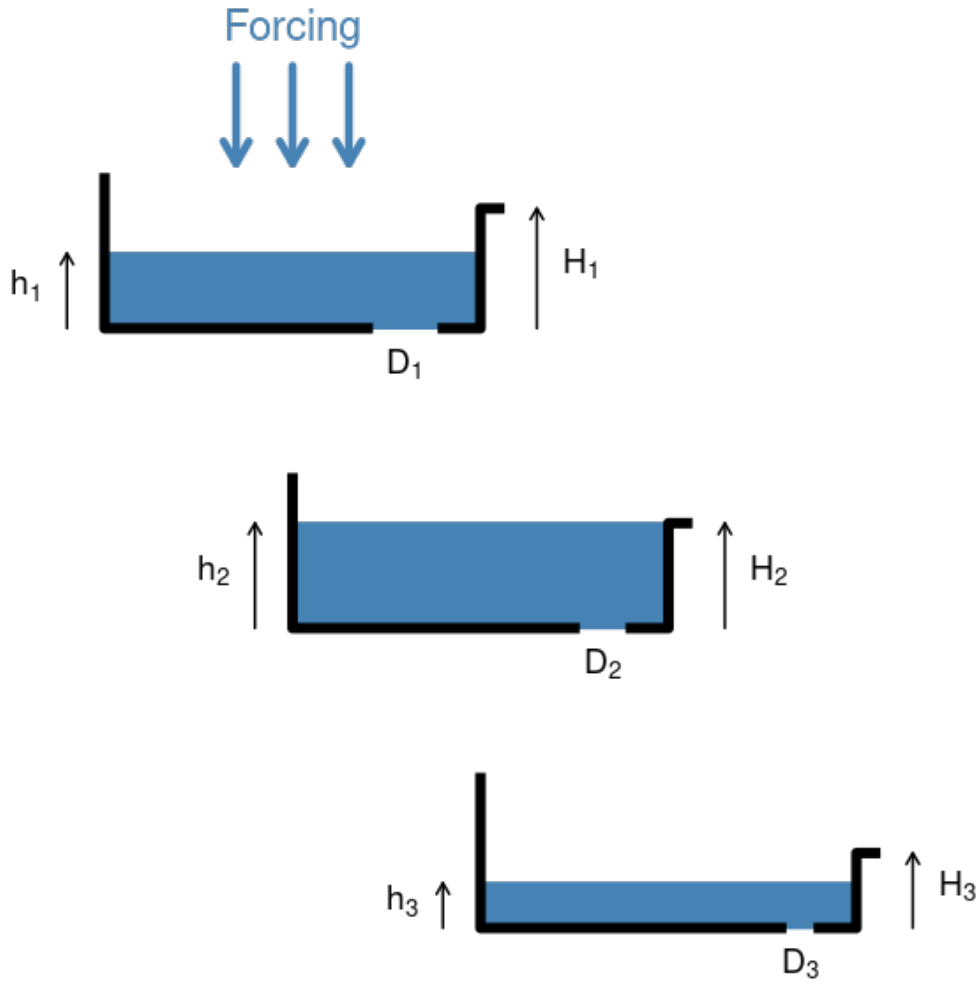
$$-2 \log L_{\mathcal{I}}(\theta^*, \alpha, \kappa) \approx \sum_{i \in \mathcal{I}} \frac{\{z_i^{\text{obs}} - (\alpha + f_i(\theta^*))\}^2}{\kappa^2 + \sigma_i^2} + \sum_{i \in \mathcal{I}} \log(\kappa^2 + \sigma_i^2),$$

- Profile out α and κ (1D numerical optimization) to give thin & prof likelihood, ▸ More details

$$L_{\mathcal{I}}^{\text{prf}}(\theta^*) := \max_{\alpha,\kappa} L_{\mathcal{I}}(\theta^*, \alpha, \kappa)$$

- I'm not claiming that this is awesome statistics (indeed, profile likelihood is a bit mysterious). But $L_{\mathcal{I}}^{\text{prf}}$ is an attainable incremental improvement on current practice.

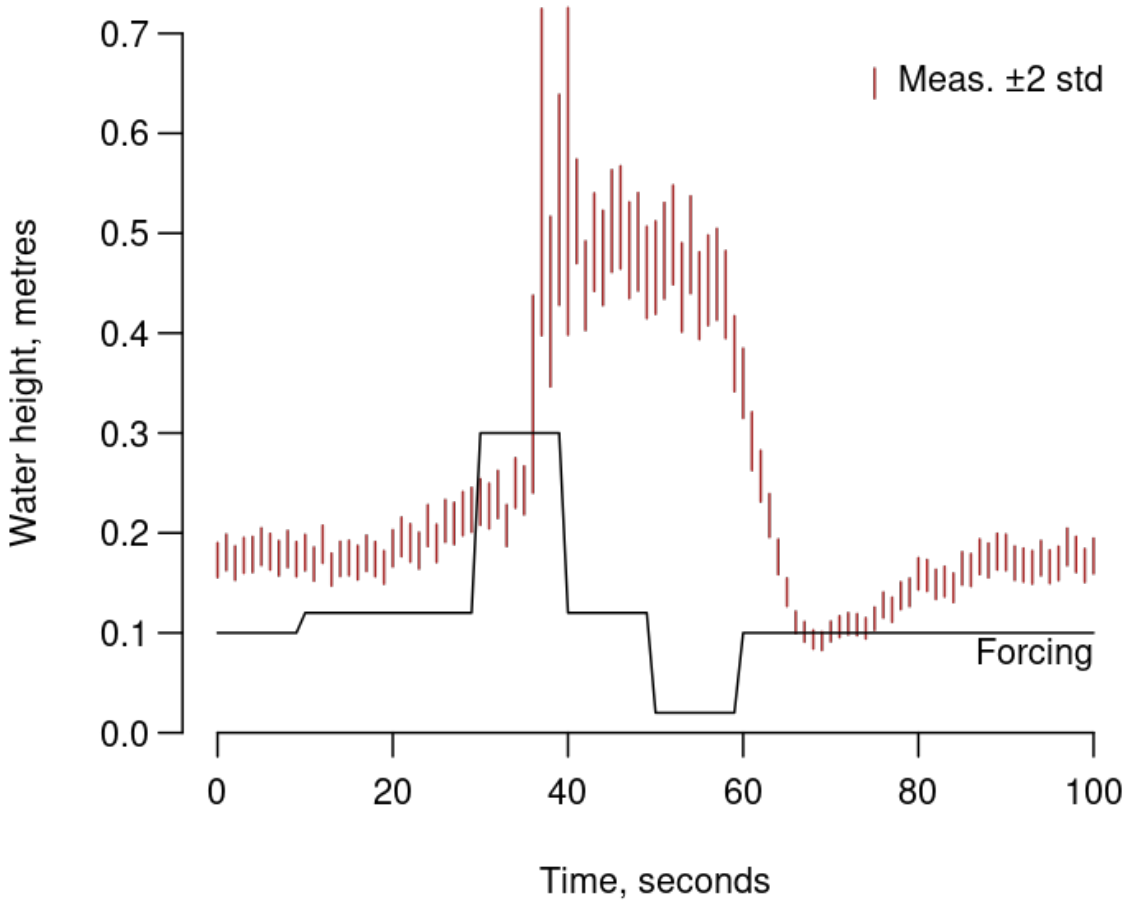
The buckets simulator



Parameters are (D_i, H_i) for each bucket. A time-series for forcing is specified, the outputs are a time-series for each h_i.

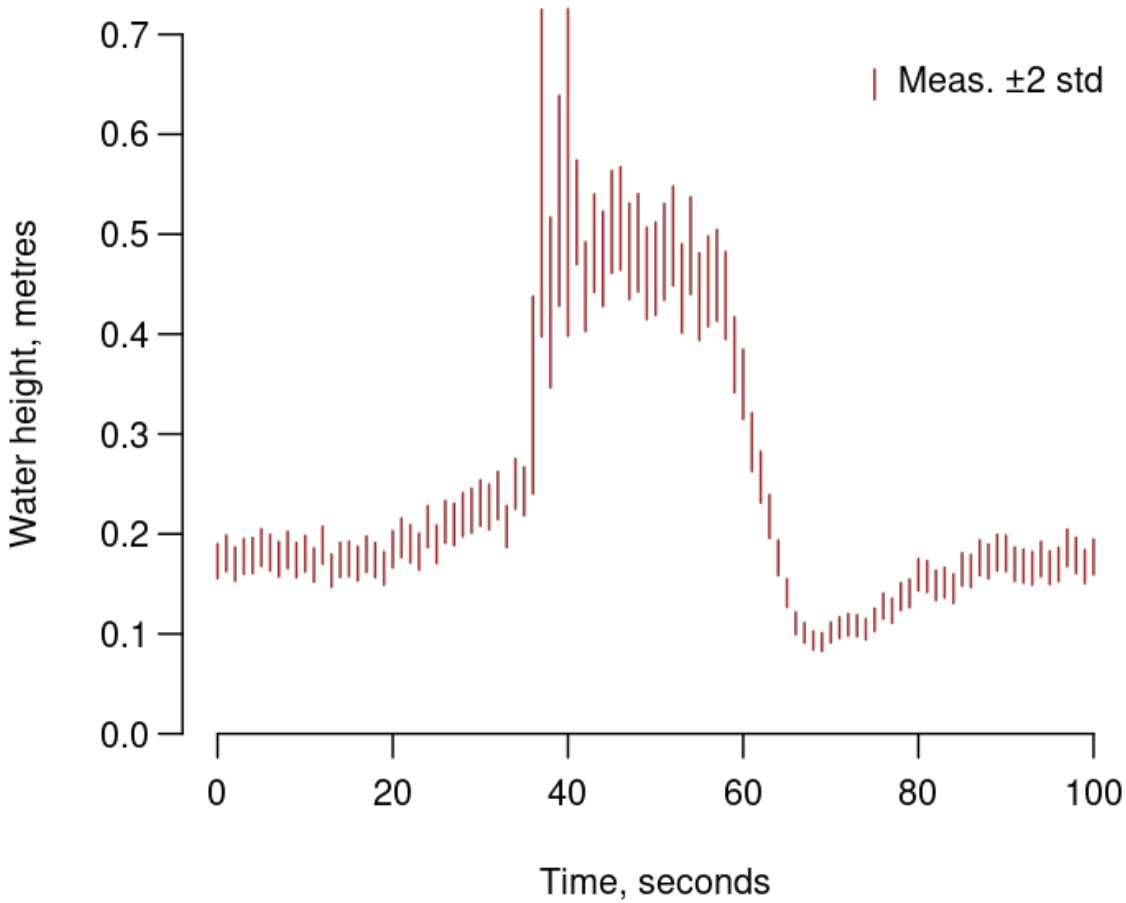
The system

Nature is running a five-bucket system, with $D_i = 0.3$, $H_i = 0.5$.
Observe h_5 with a known state-dependent measurement error.



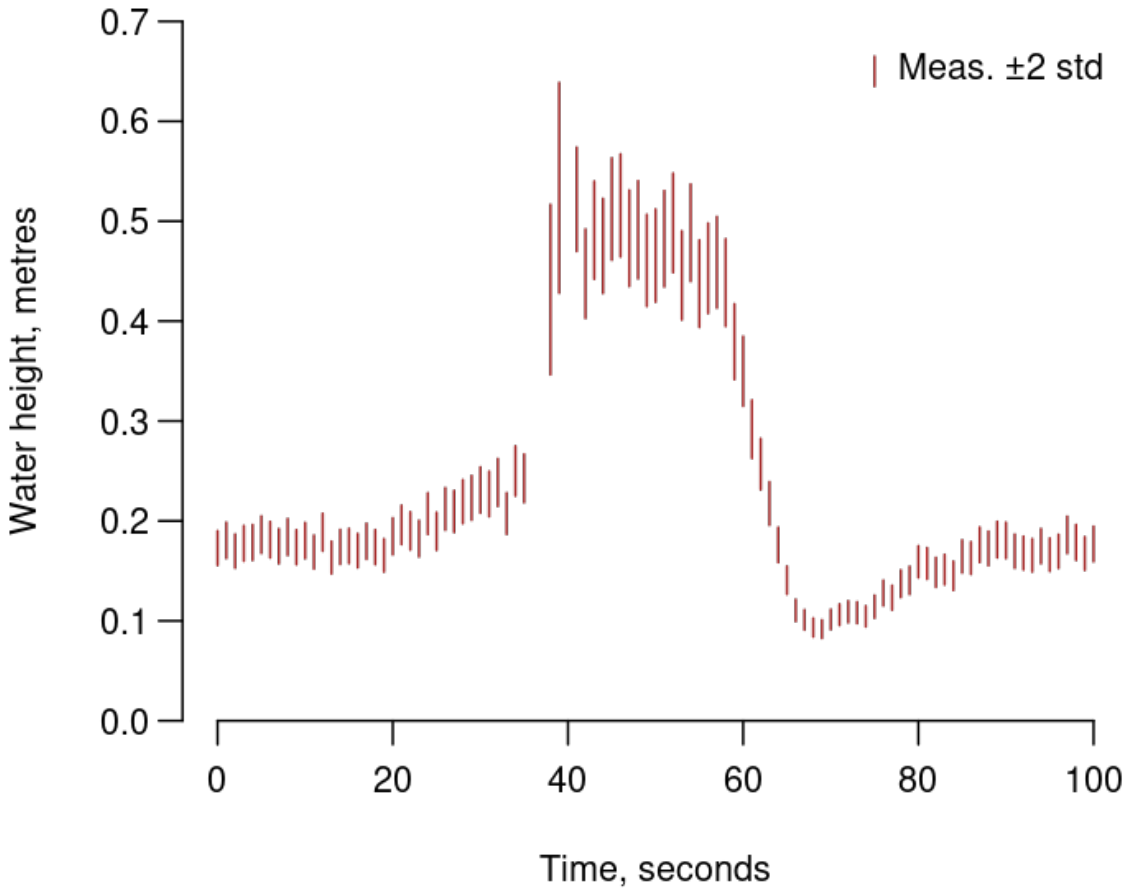
Thinning = 'feature extraction'

All of the observations



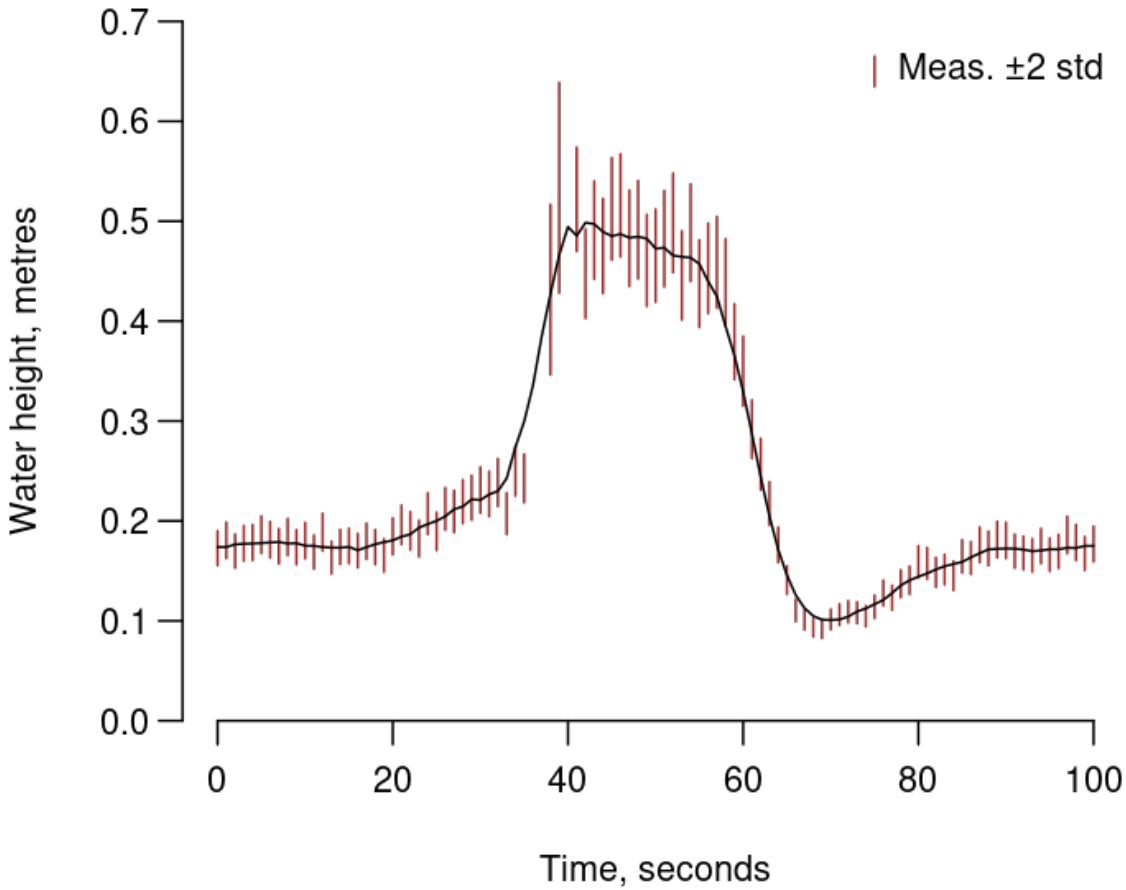
Thinning = ‘feature extraction’

Drop the noisiest



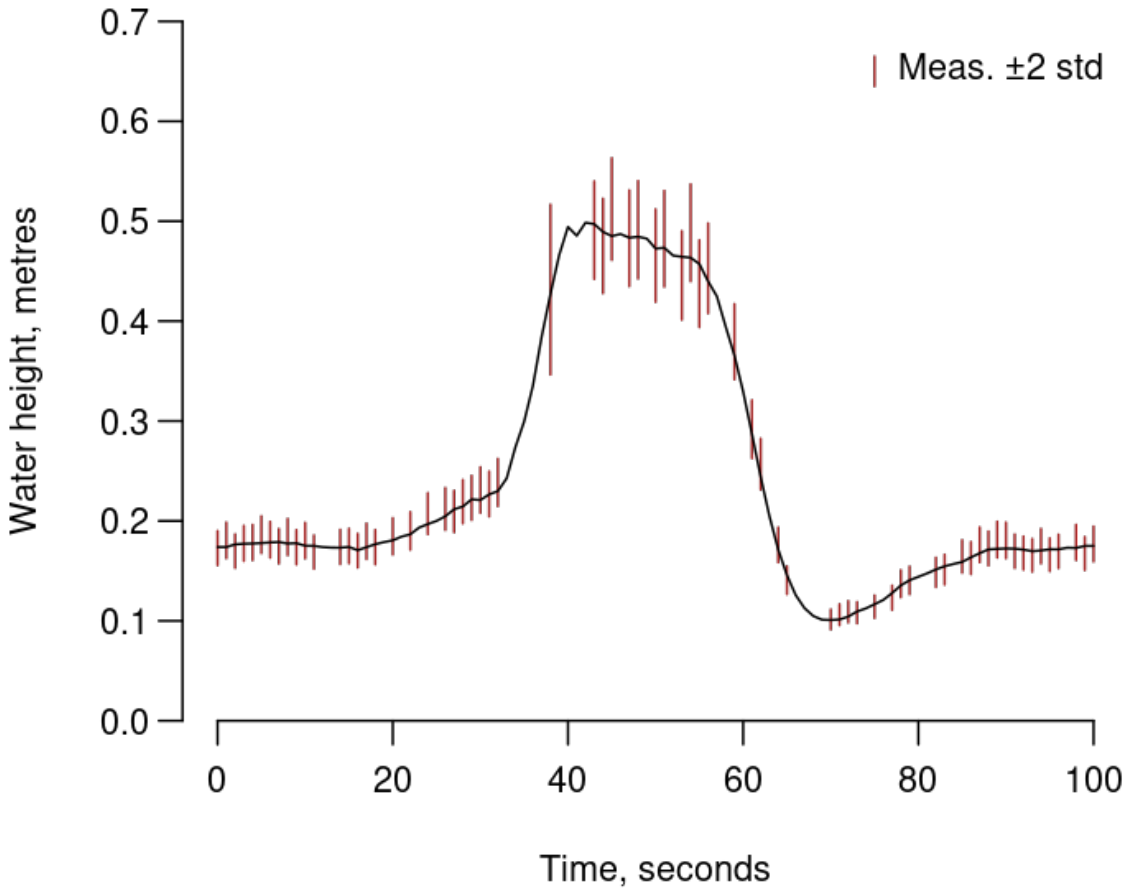
Thinning = 'feature extraction'

Add a moving average



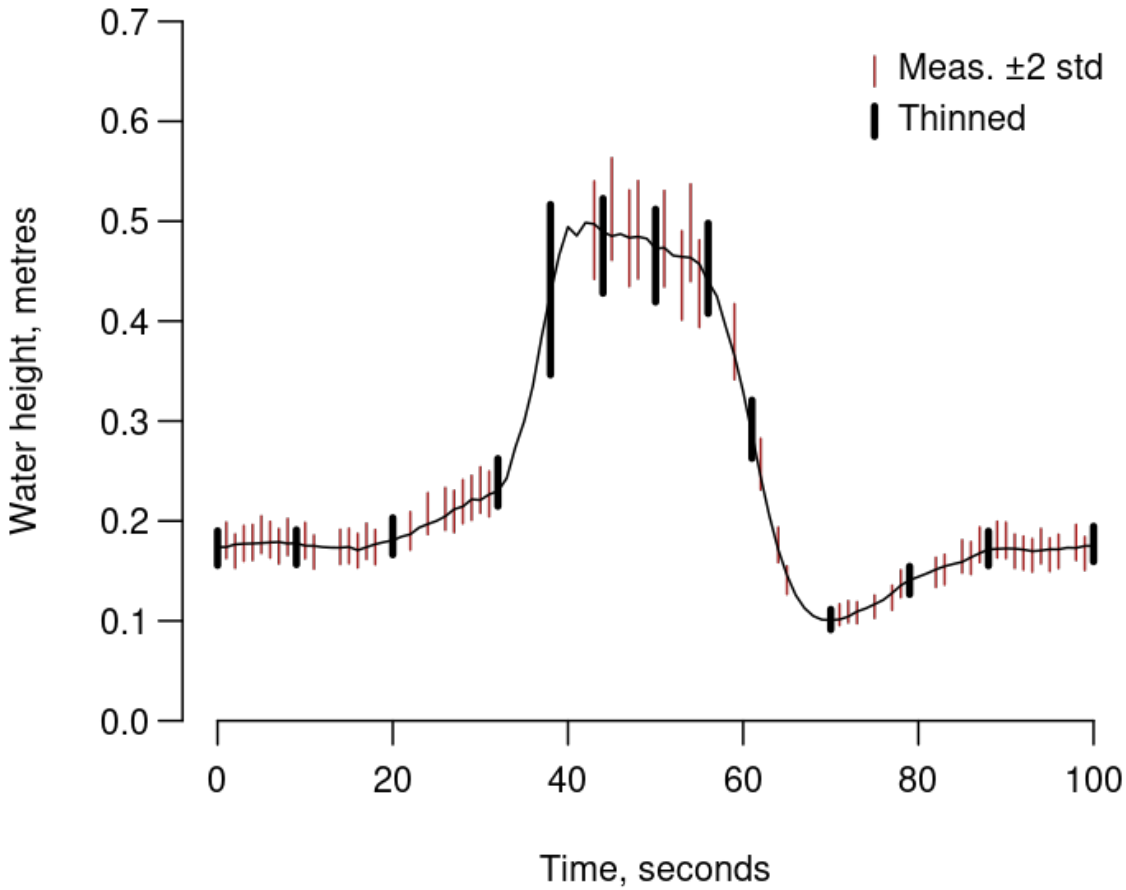
Thinning = ‘feature extraction’

Drop the outliers



Thinning = 'feature extraction'

Thin the survivors

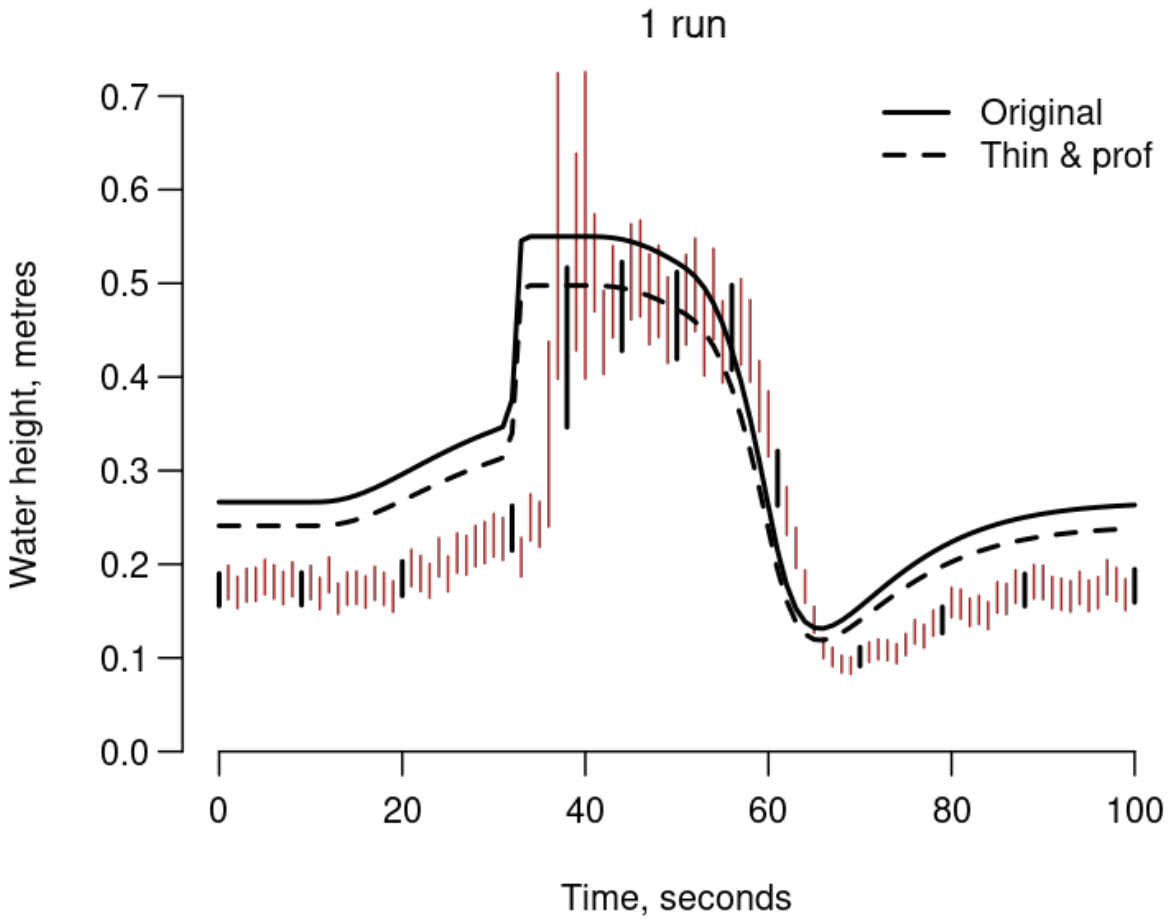


Our simulator

We're running a three-bucket simulator, matching our h_3 to observations of nature's h_5. Here are the initial stages of a Downhill Simplex optimizer.

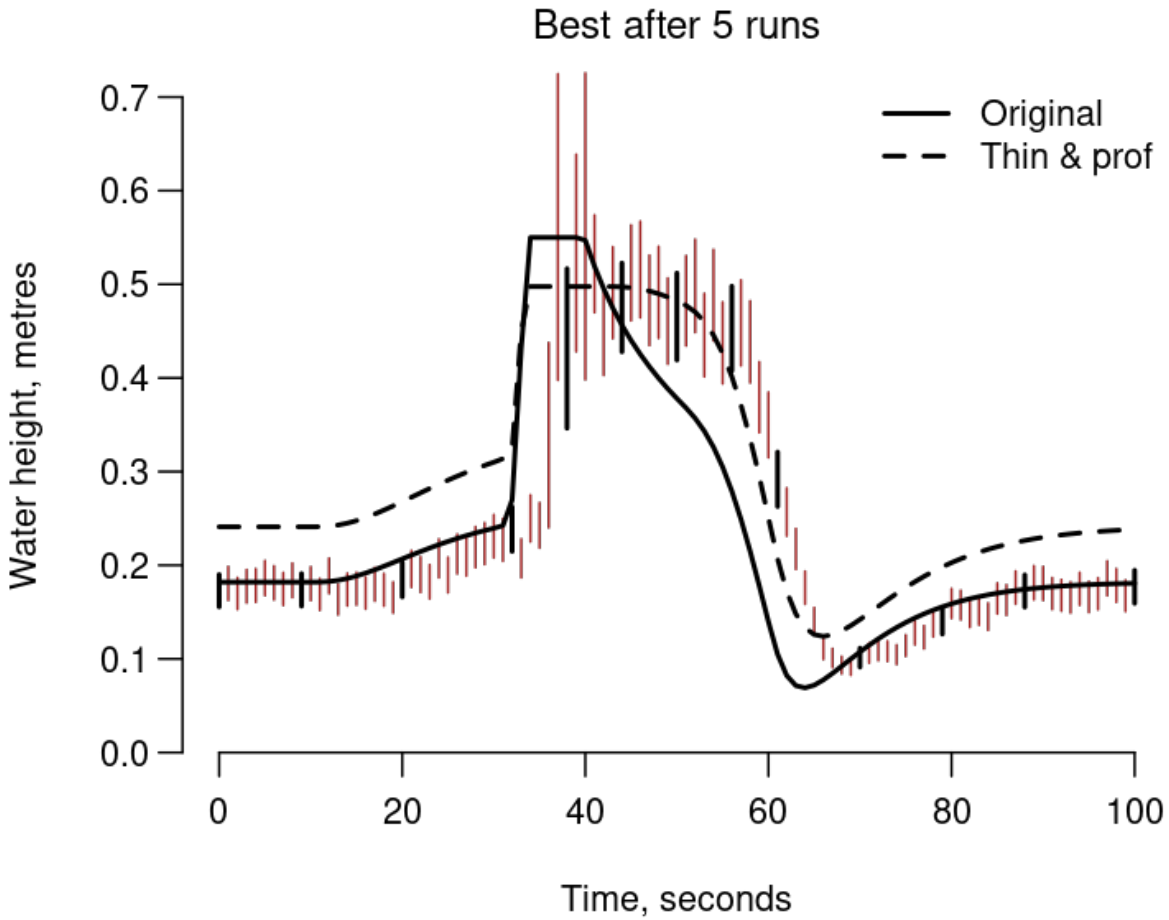
Our simulator

We're running a three-bucket simulator, matching our h_3 to observations of nature's h_5. Here are the initial stages of a Downhill Simplex optimizer.



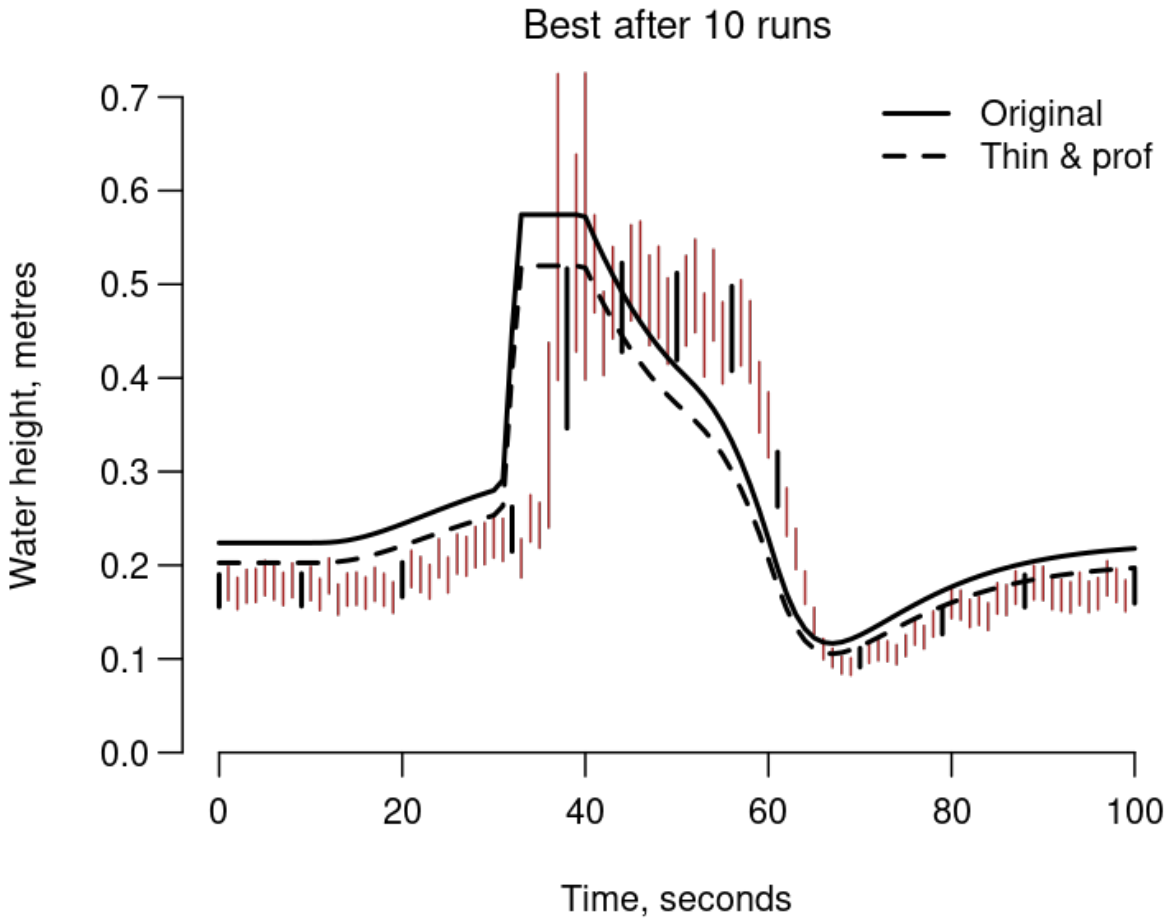
Our simulator

We're running a three-bucket simulator, matching our h_3 to observations of nature's h_5. Here are the initial stages of a Downhill Simplex optimizer.



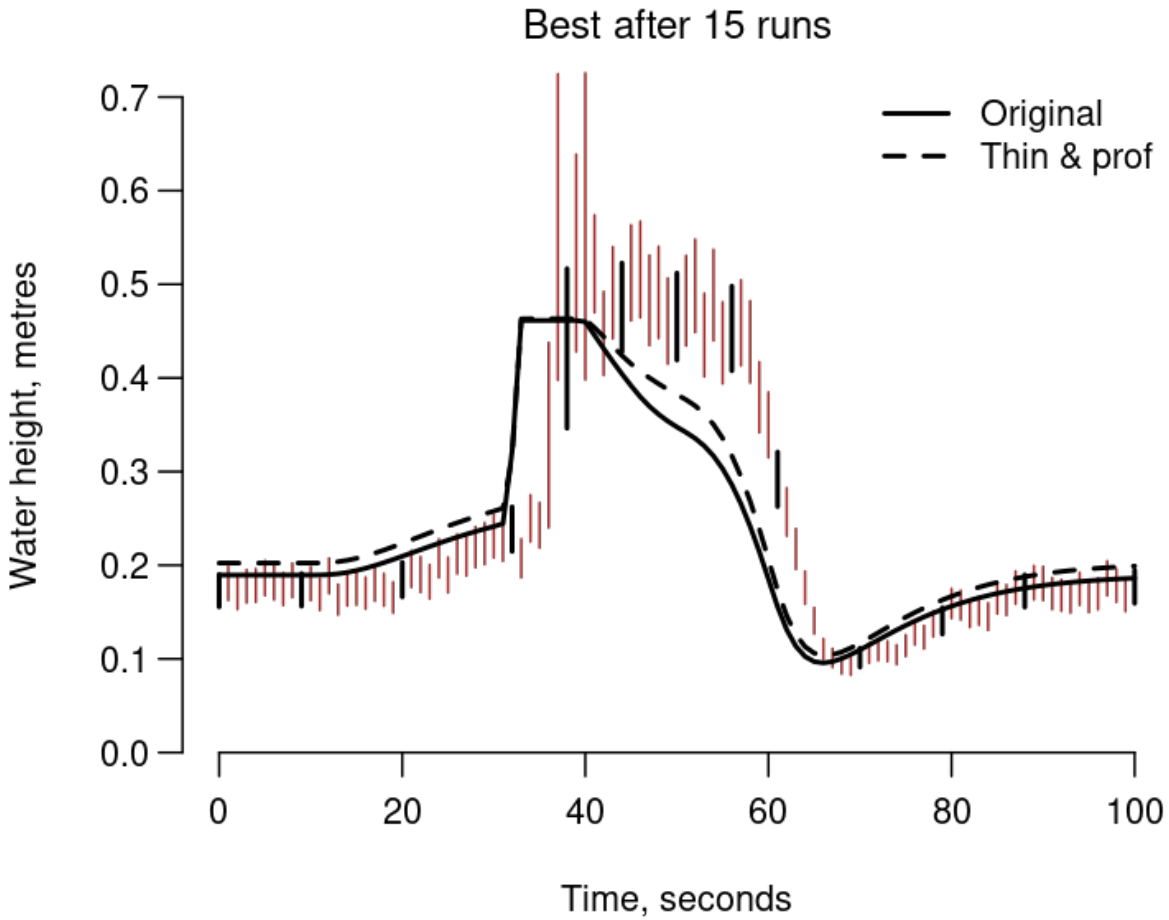
Our simulator

We're running a three-bucket simulator, matching our h_3 to observations of nature's h_5. Here are the initial stages of a Downhill Simplex optimizer.



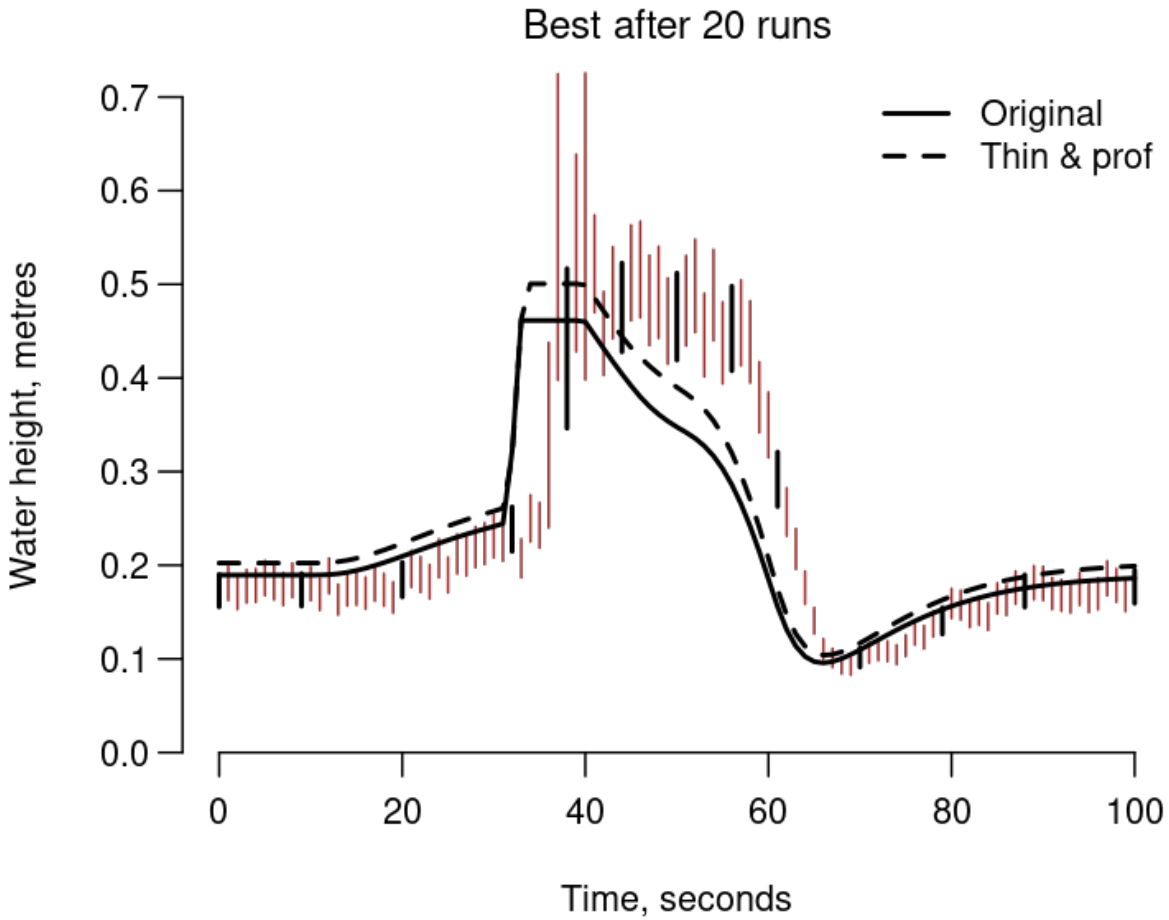
Our simulator

We're running a three-bucket simulator, matching our h_3 to observations of nature's h_5. Here are the initial stages of a Downhill Simplex optimizer.



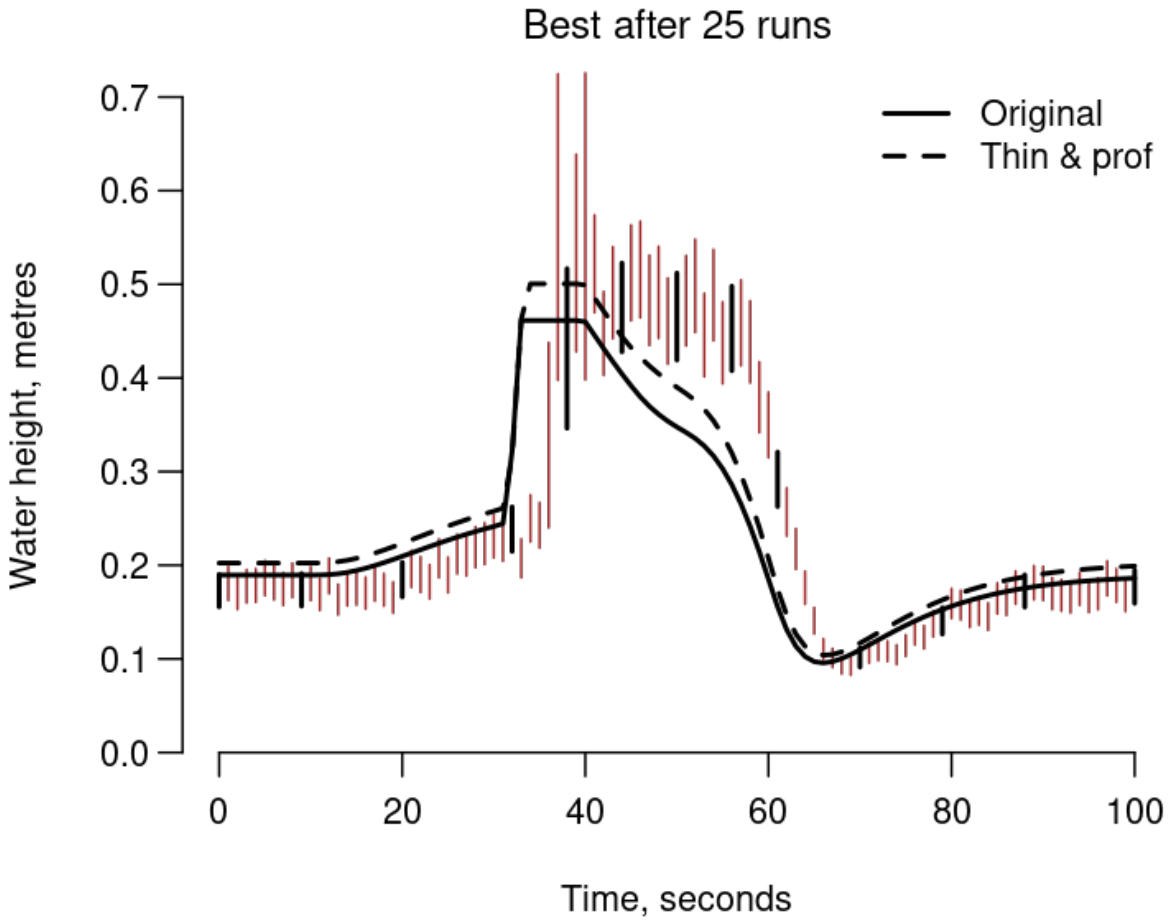
Our simulator

We're running a three-bucket simulator, matching our h_3 to observations of nature's h_5. Here are the initial stages of a Downhill Simplex optimizer.



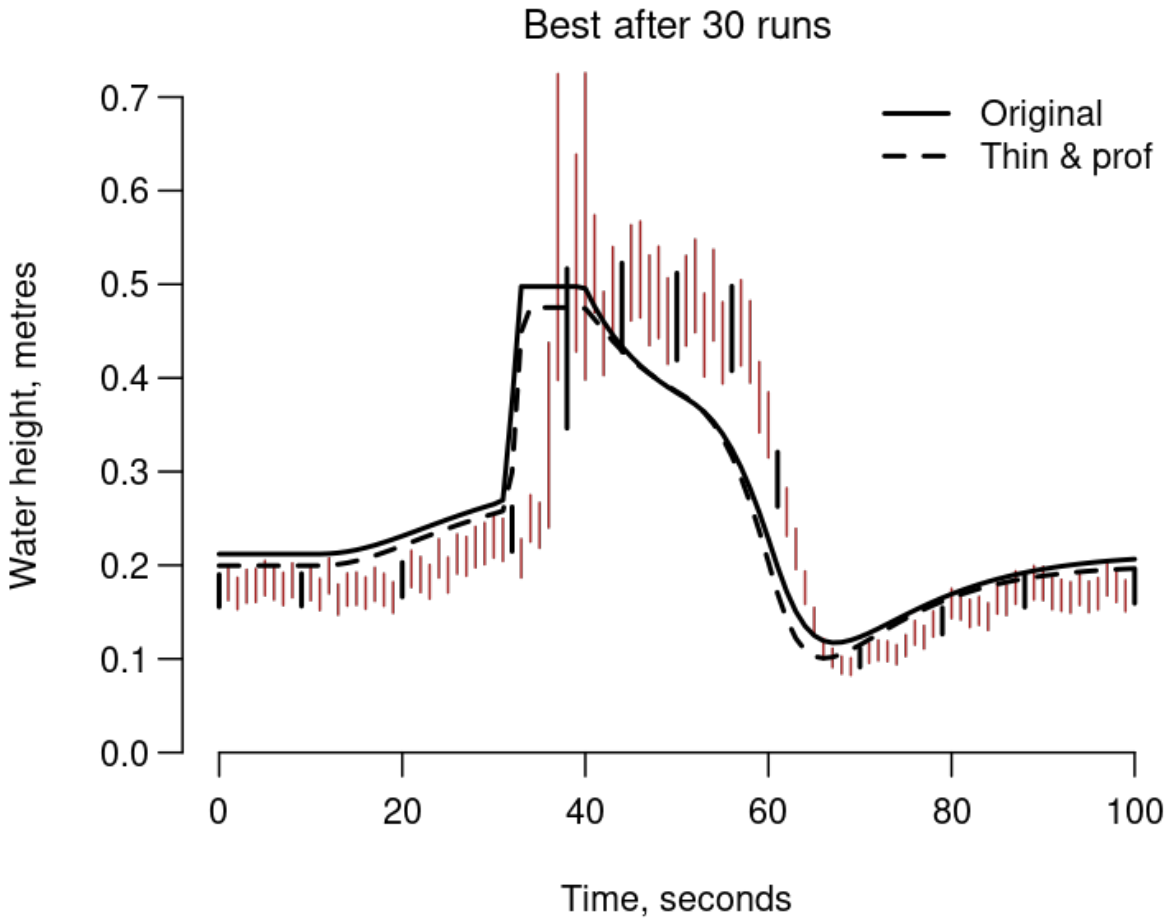
Our simulator

We're running a three-bucket simulator, matching our h_3 to observations of nature's h_5. Here are the initial stages of a Downhill Simplex optimizer.



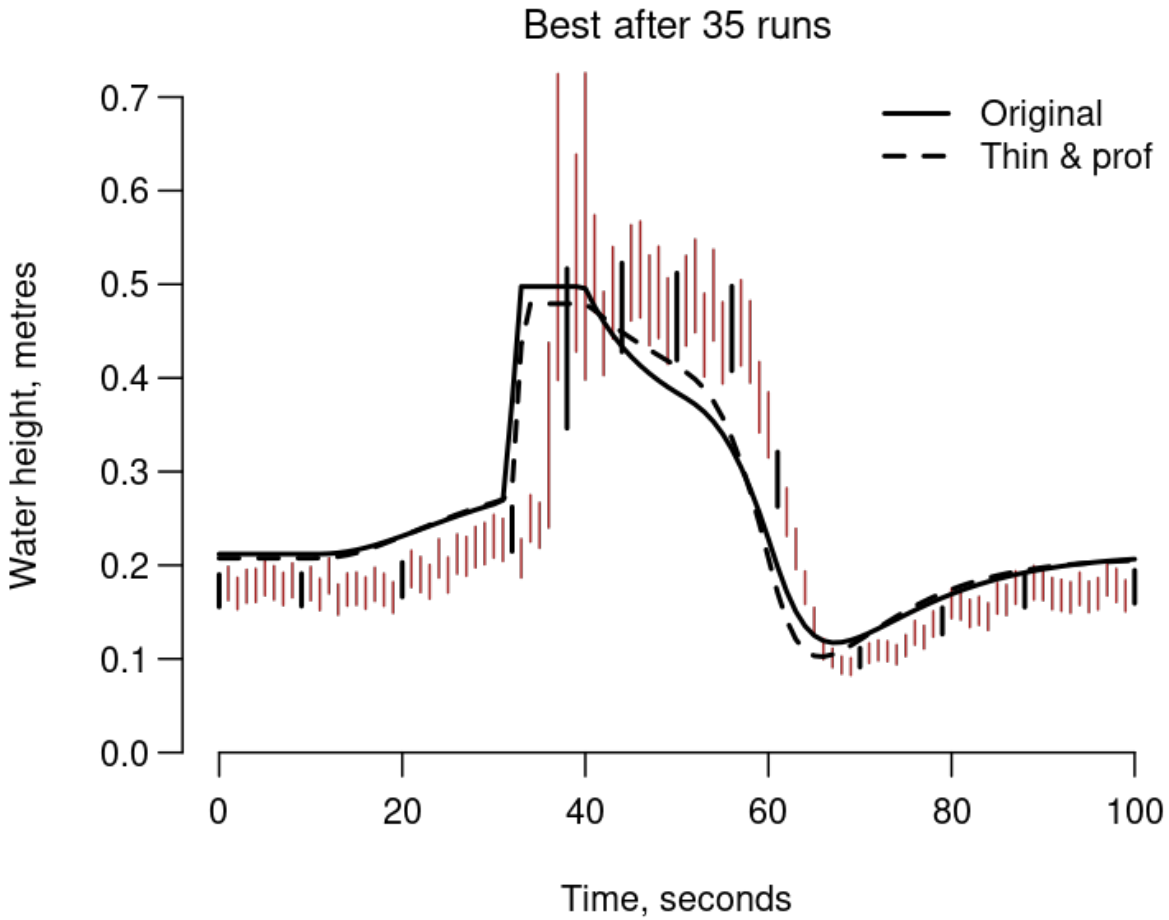
Our simulator

We're running a three-bucket simulator, matching our h_3 to observations of nature's h_5. Here are the initial stages of a Downhill Simplex optimizer.



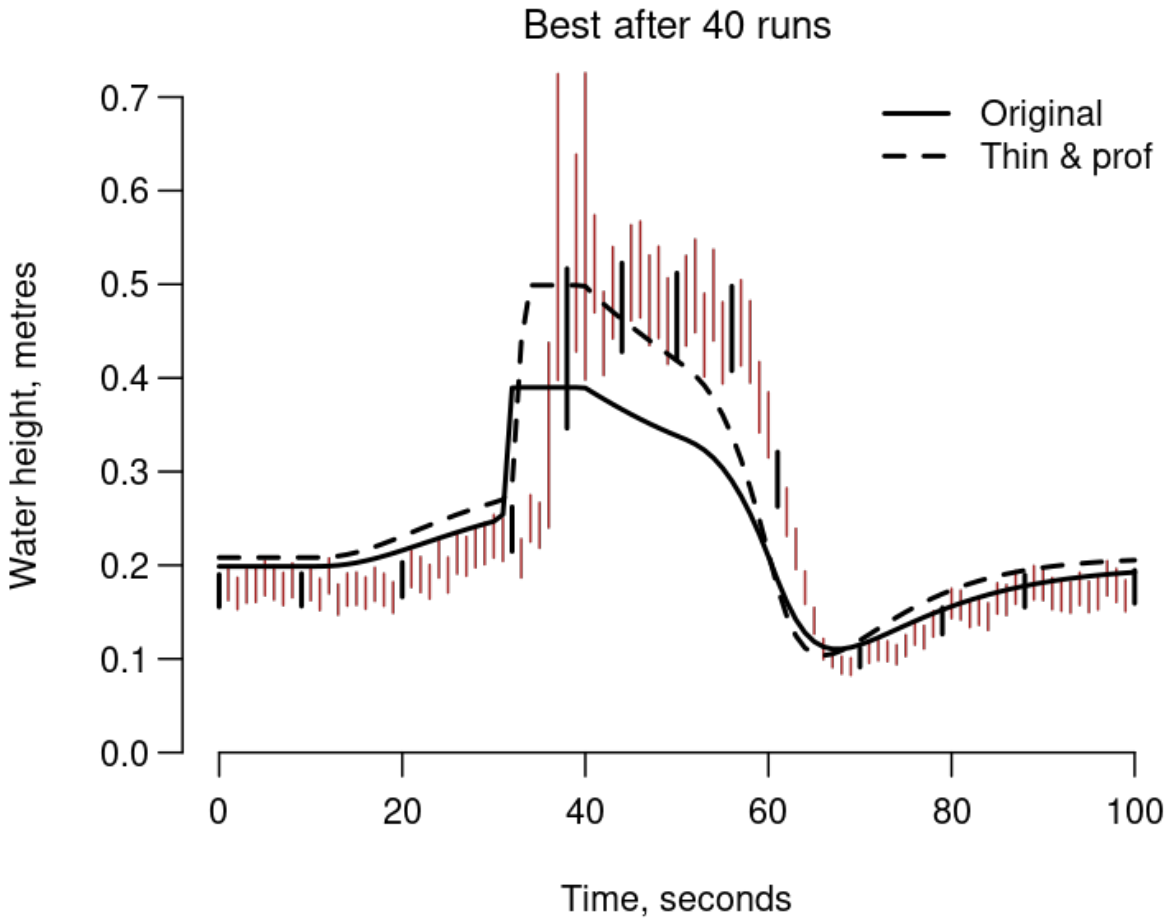
Our simulator

We're running a three-bucket simulator, matching our h_3 to observations of nature's h_5. Here are the initial stages of a Downhill Simplex optimizer.



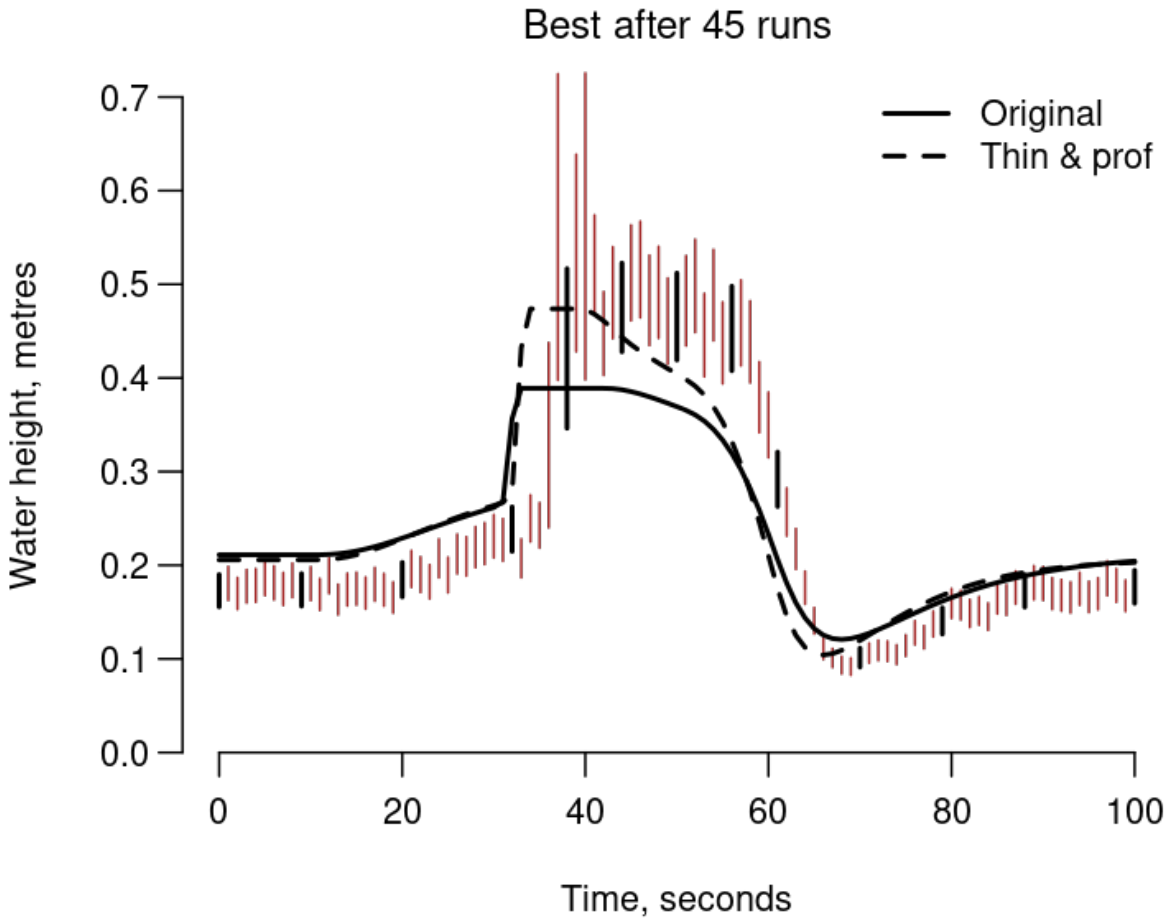
Our simulator

We're running a three-bucket simulator, matching our h_3 to observations of nature's h_5. Here are the initial stages of a Downhill Simplex optimizer.



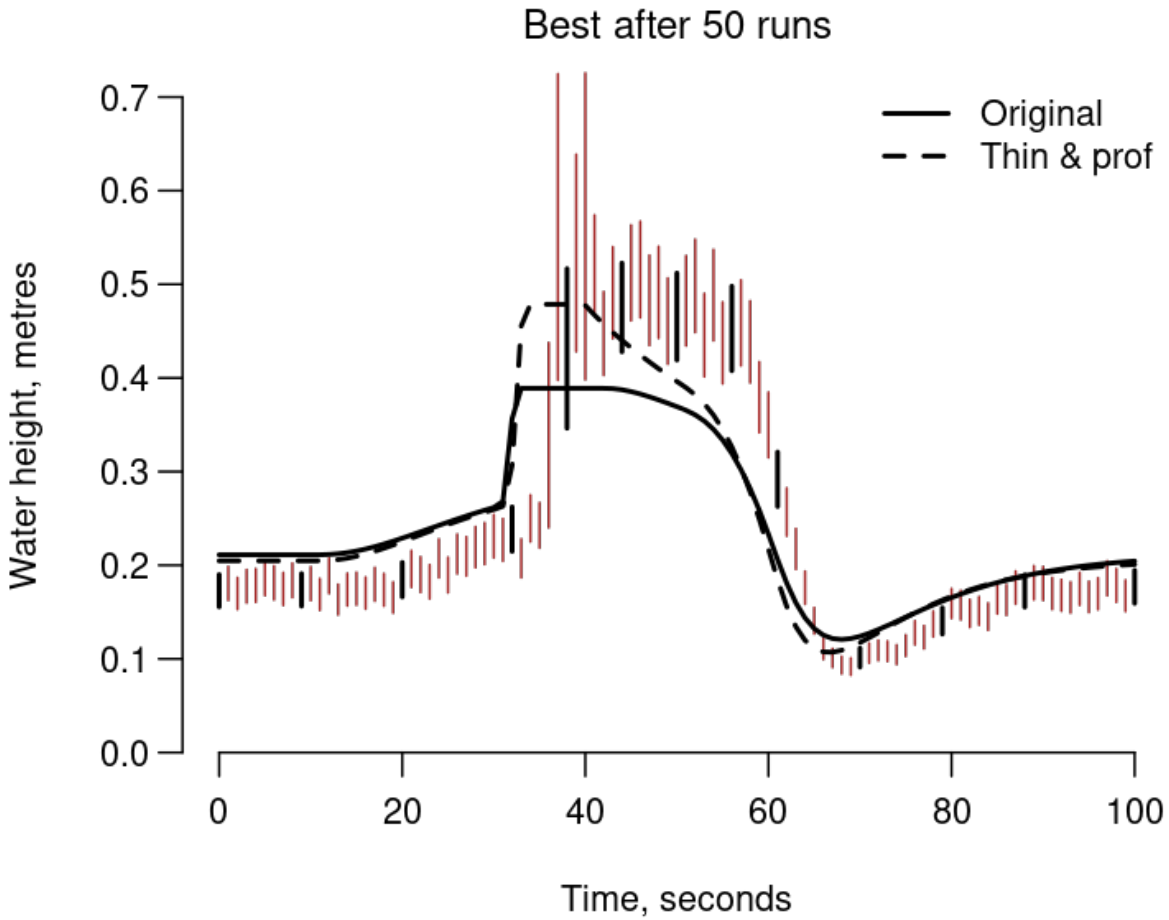
Our simulator

We're running a three-bucket simulator, matching our h_3 to observations of nature's h_5. Here are the initial stages of a Downhill Simplex optimizer.



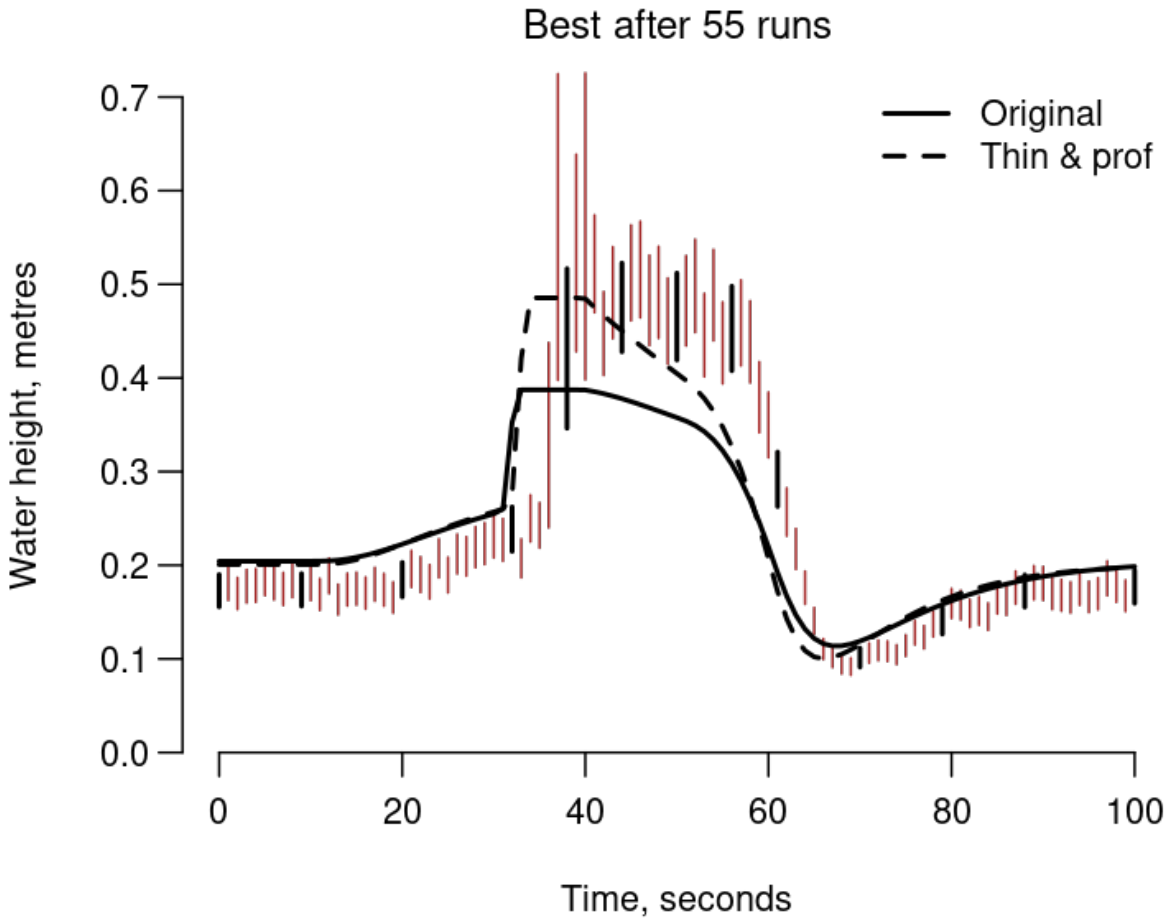
Our simulator

We're running a three-bucket simulator, matching our h_3 to observations of nature's h_5. Here are the initial stages of a Downhill Simplex optimizer.



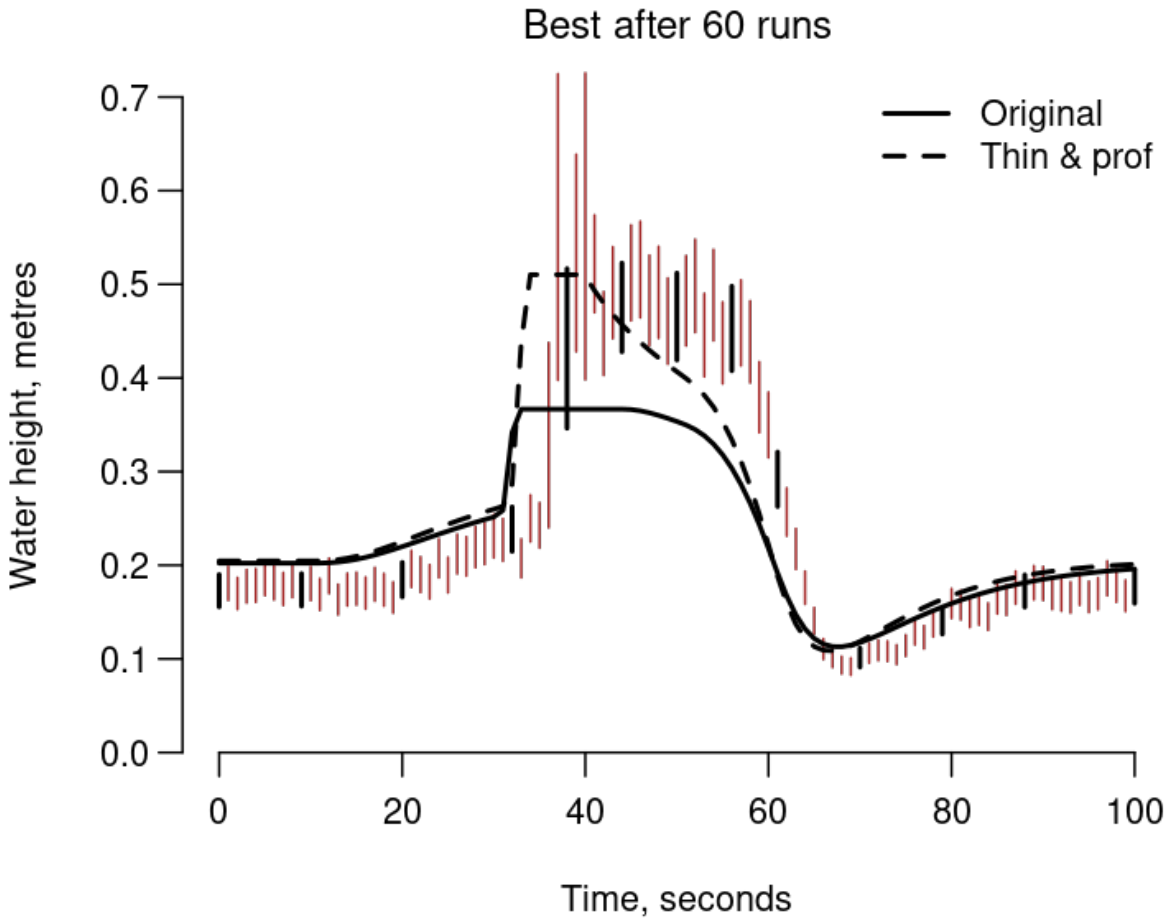
Our simulator

We're running a three-bucket simulator, matching our h_3 to observations of nature's h_5. Here are the initial stages of a Downhill Simplex optimizer.



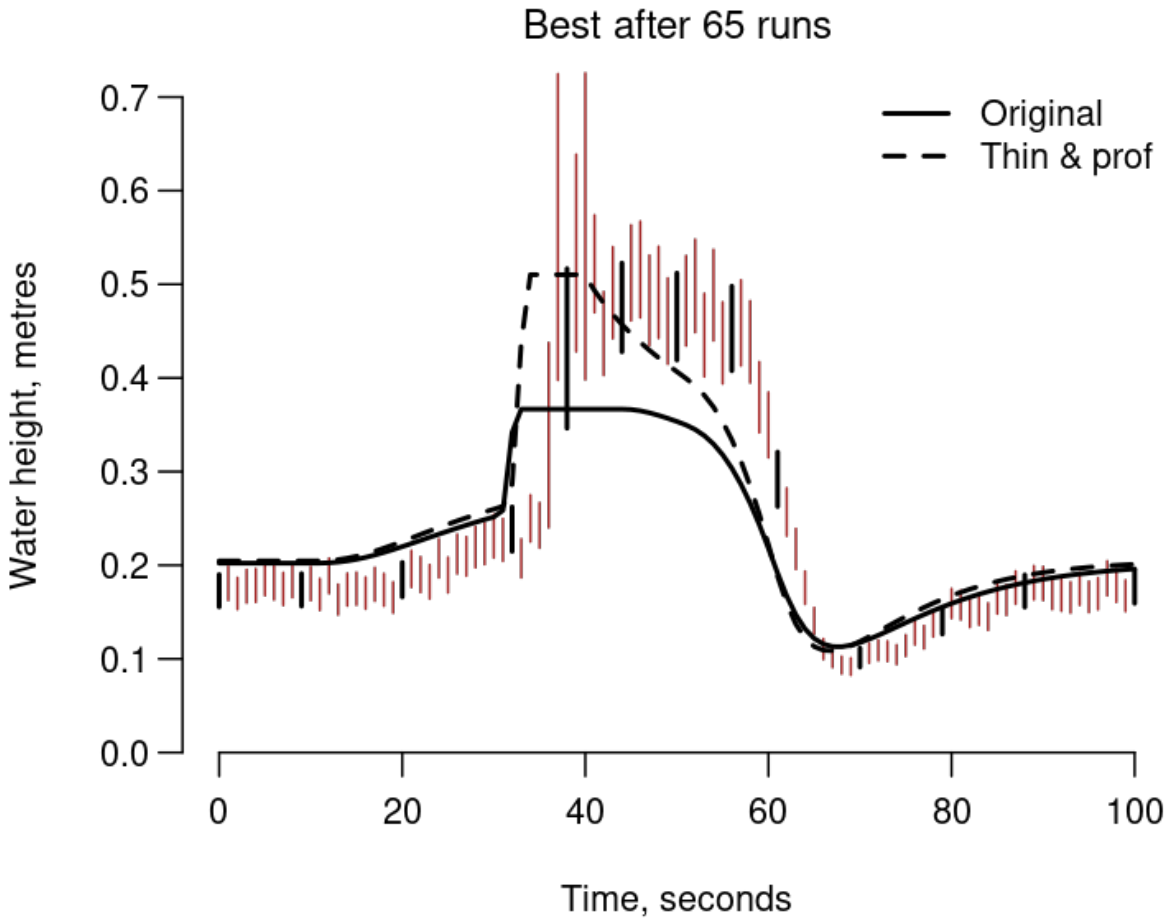
Our simulator

We're running a three-bucket simulator, matching our h_3 to observations of nature's h_5. Here are the initial stages of a Downhill Simplex optimizer.



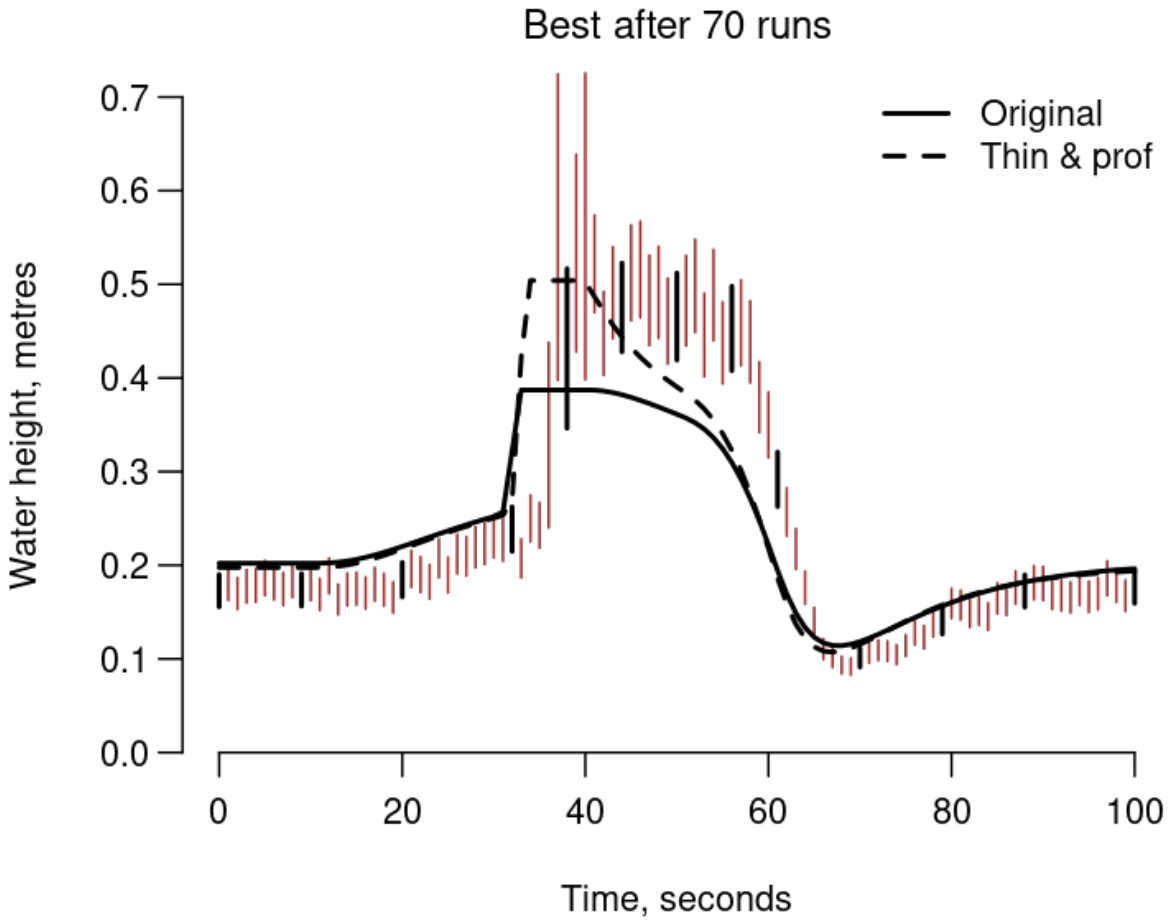
Our simulator

We're running a three-bucket simulator, matching our h_3 to observations of nature's h_5. Here are the initial stages of a Downhill Simplex optimizer.



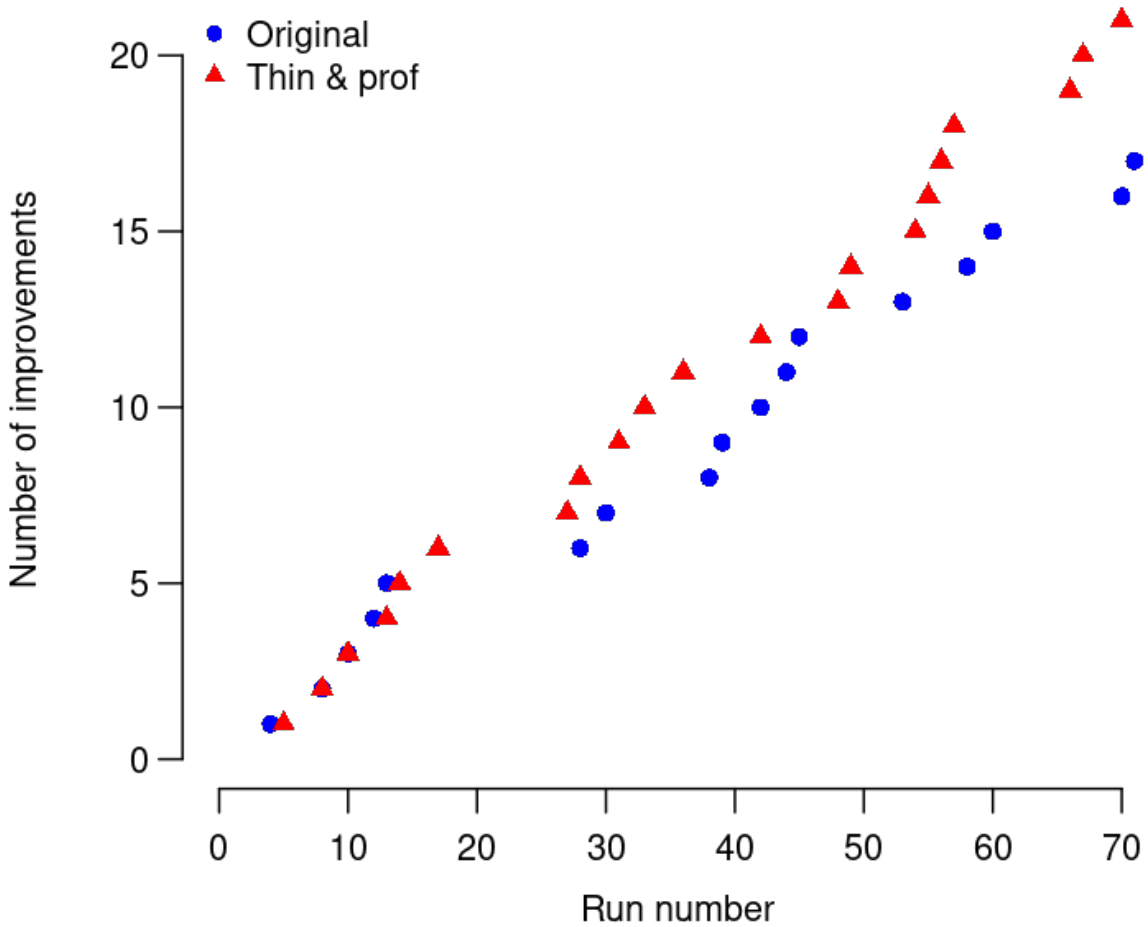
Our simulator

We're running a three-bucket simulator, matching our h_3 to observations of nature's h_5. Here are the initial stages of a Downhill Simplex optimizer.



What's going on?

The original likelihood function is lumpy (Rougier, 2013), and it is hard for the optimiser to make progress. Not so the thin & prof likelihood:



Where next?

The thin & prof log-likelihood function ought to be smooth-ish, especially near to the global maximum. This seems like a good candidate for Bayesian optimization, to squeeze out a better fit than the best so far.

Bayesian optimization

- Let m and v be the current expectation and variance functions of the GP emulator of $\ell := -2 \log L_{\mathcal{J}}^{\mathsf{prf}}$; should be smooth-ish functions of θ.

Bayesian optimization

- Let m and v be the current expectation and variance functions of the GP emulator of $\ell := -2 \log L_{\mathcal{J}}^{\text{prf}}$; should be smooth-ish functions of θ.

- Let ℓ^{best} be the best (smallest) value found so far. We choose the next run to be at the θ which minimizes the expected value of the 'improvement'

$$\lambda(\ell_\theta) := \begin{cases} \ell_\theta & \ell_\theta < \ell^{\text{best}} \\ \ell^{\text{best}} & \ell_\theta \geq \ell^{\text{best}} \end{cases}$$

  as proposed in Osborne et al. (2009).

Bayesian optimization

- Let m and v be the current expectation and variance functions of the GP emulator of $\ell := -2 \log L_{\mathcal{J}}^{\text{prf}}$; should be smooth-ish functions of θ.

- Let ℓ^{best} be the best (smallest) value found so far. We choose the next run to be at the θ which minimizes the expected value of the 'improvement'

$$\lambda(\ell_\theta) := \begin{cases} \ell_\theta & \ell_\theta < \ell^{\text{best}} \\ \ell^{\text{best}} & \ell_\theta \geq \ell^{\text{best}} \end{cases}$$

  as proposed in Osborne et al. (2009).

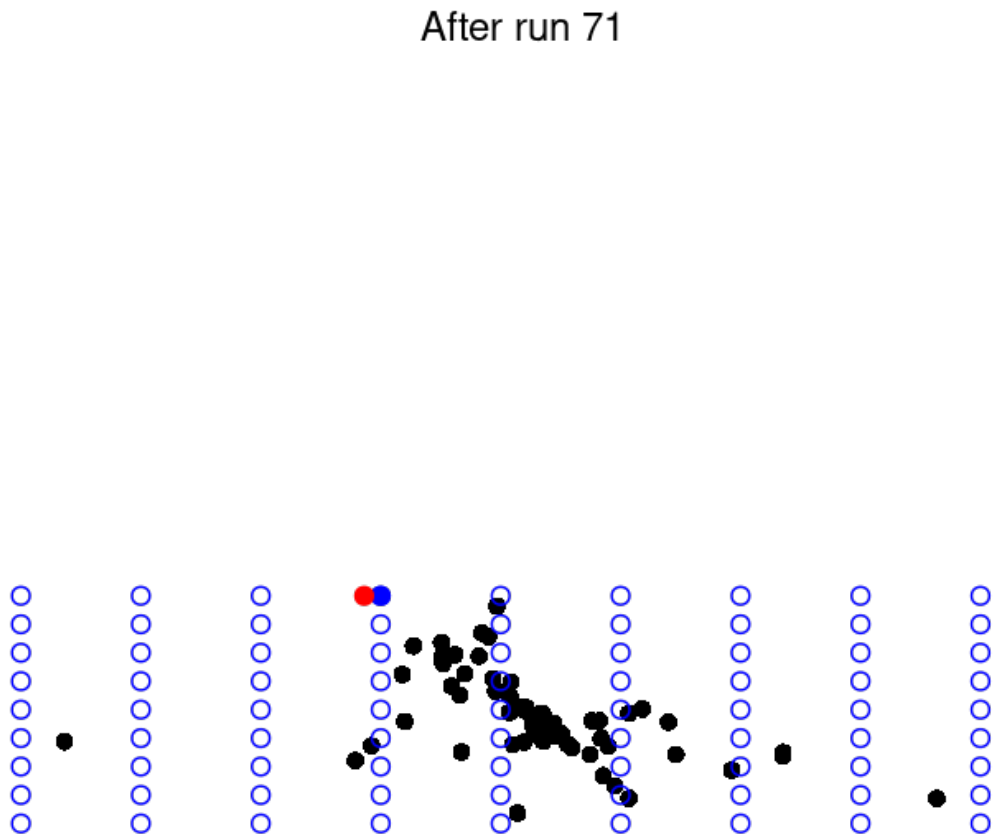
- Some algebra shows that

$$\mathbb{E}\{\lambda(L_\theta)\} = \ell^{\text{best}} + (m(\theta) - \ell^{\text{best}})\Phi(\ell^{\text{best}}) - v(\theta)\phi(\ell^{\text{best}}),$$

  where L_θ is the unknown value of ℓ_θ, Φ and ϕ are the Gaussian distribution function and density function, evaluated with $m(\theta)$ and $v(\theta)$.

Bayesian optimization (cont)

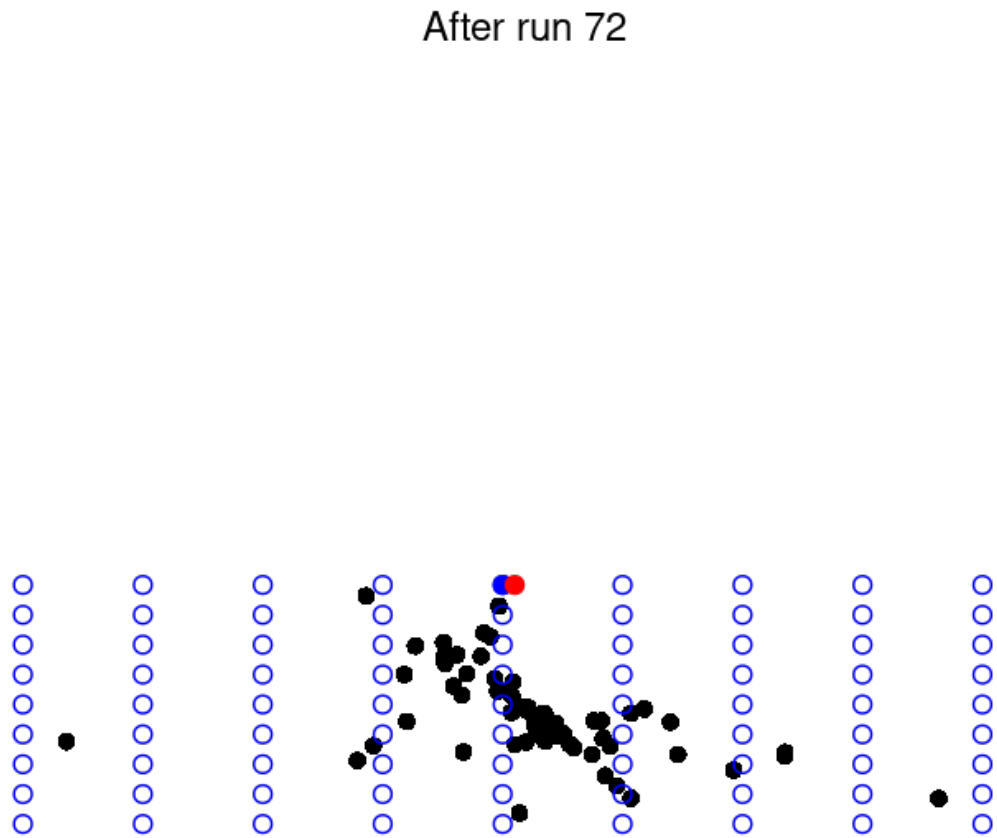
Simple adaptive search:



▸ More details

Bayesian optimization (cont)

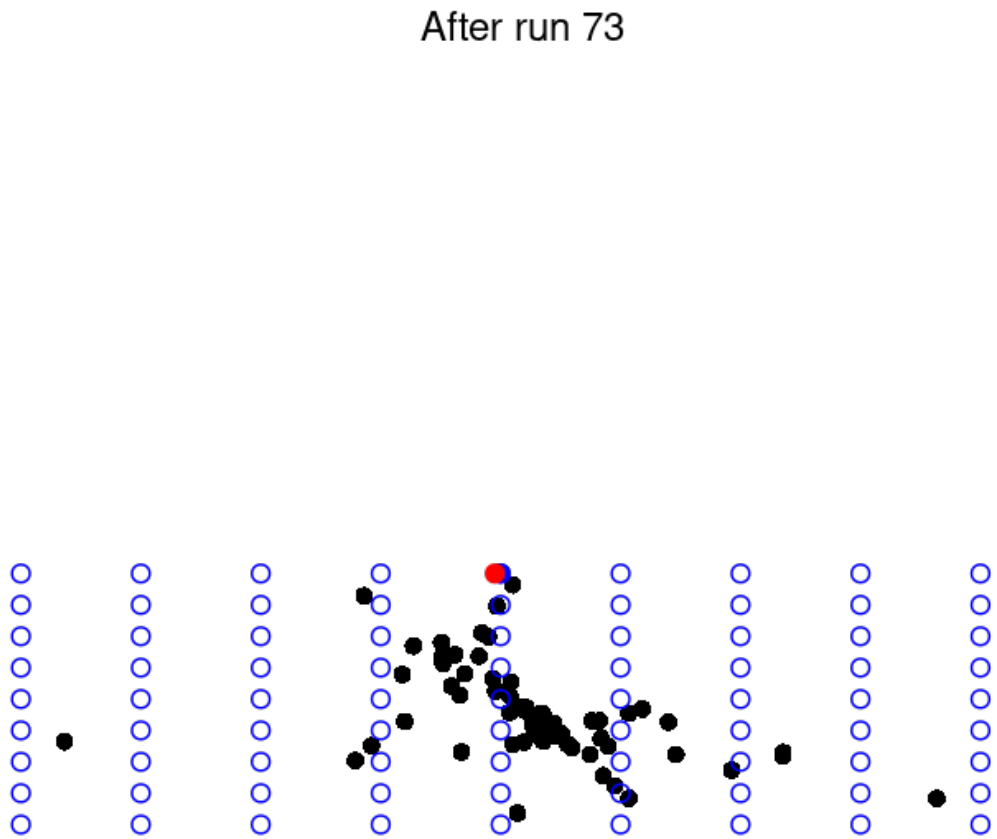
Simple adaptive search:



▸ More details

Bayesian optimization (cont)

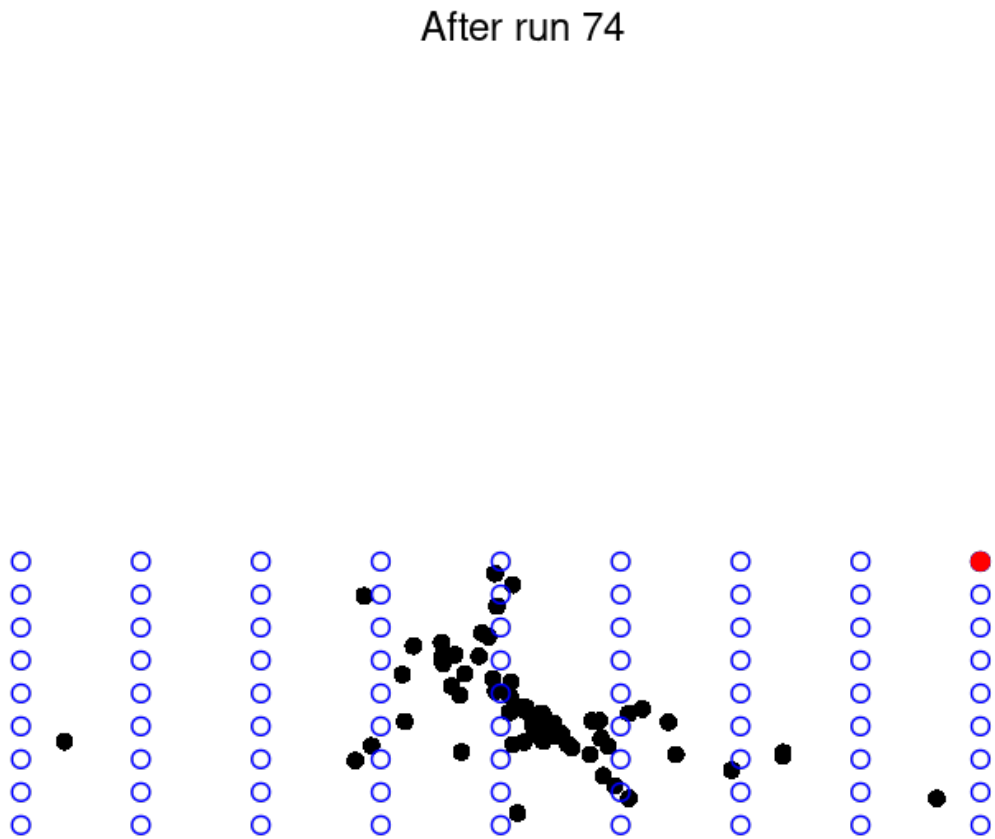
Simple adaptive search:



▸ More details

Bayesian optimization (cont)

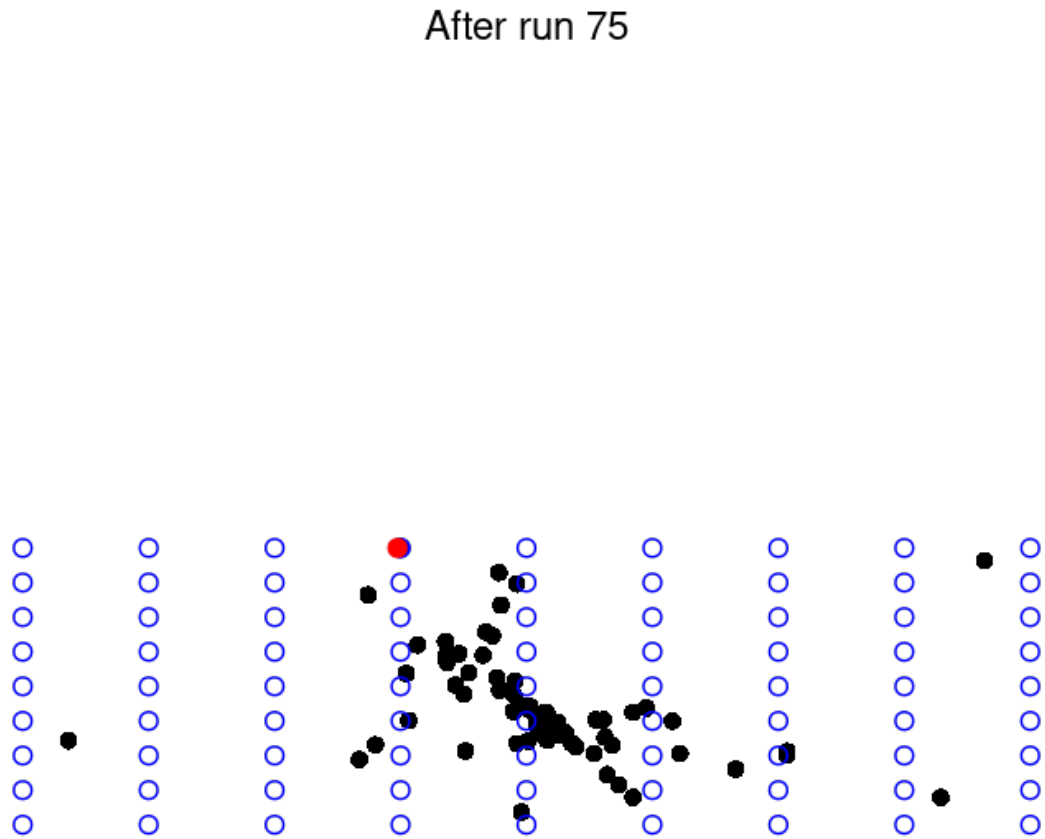
Simple adaptive search:



▸ More details

Bayesian optimization (cont)

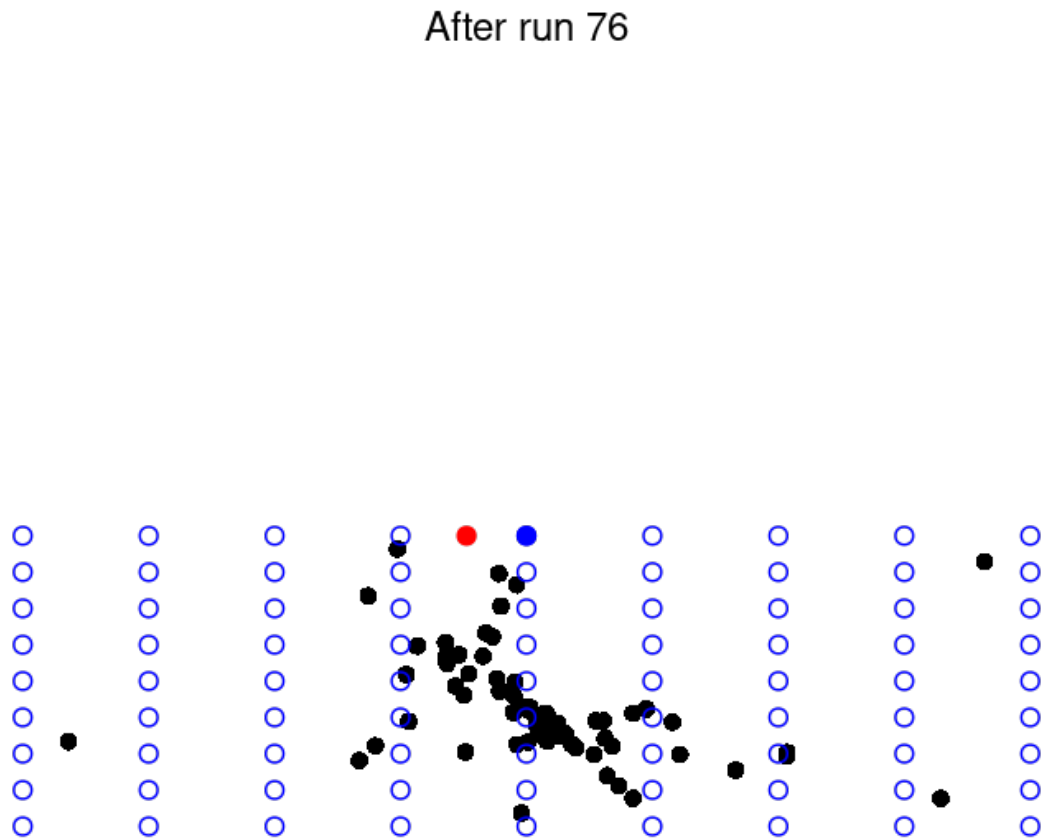
Simple adaptive search:



▸ More details

Bayesian optimization (cont)

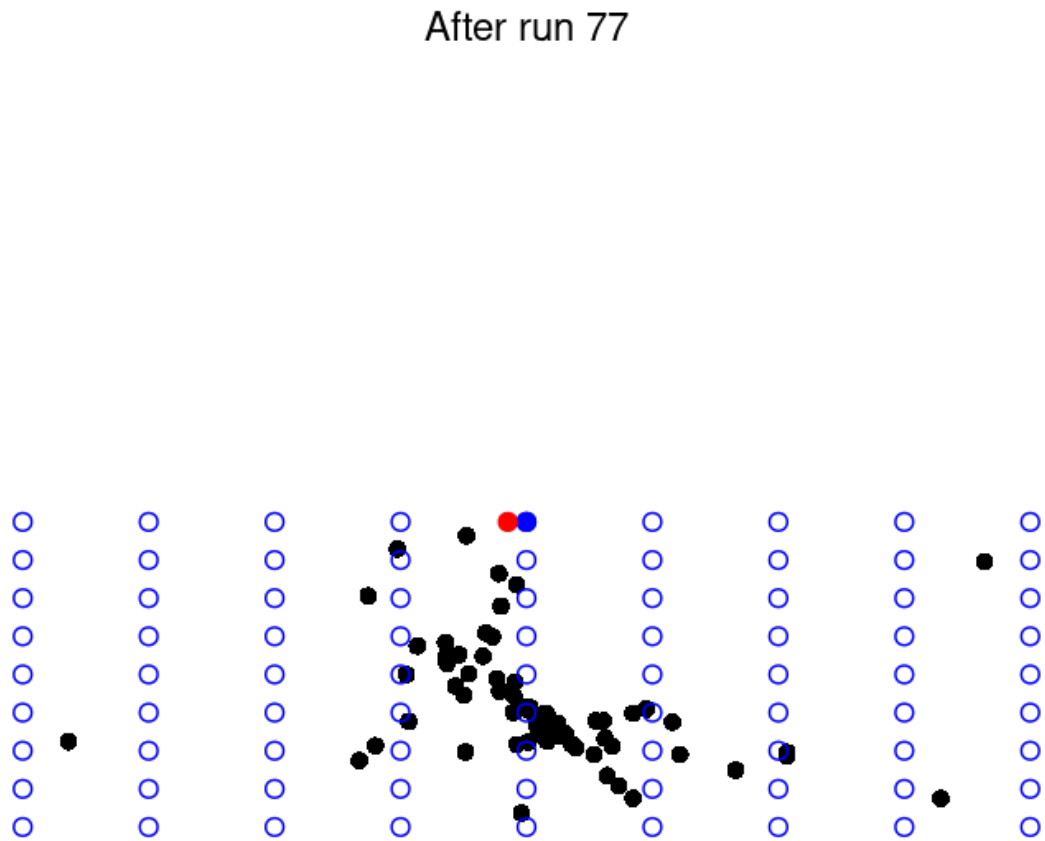
Simple adaptive search:



▸ More details

Bayesian optimization (cont)

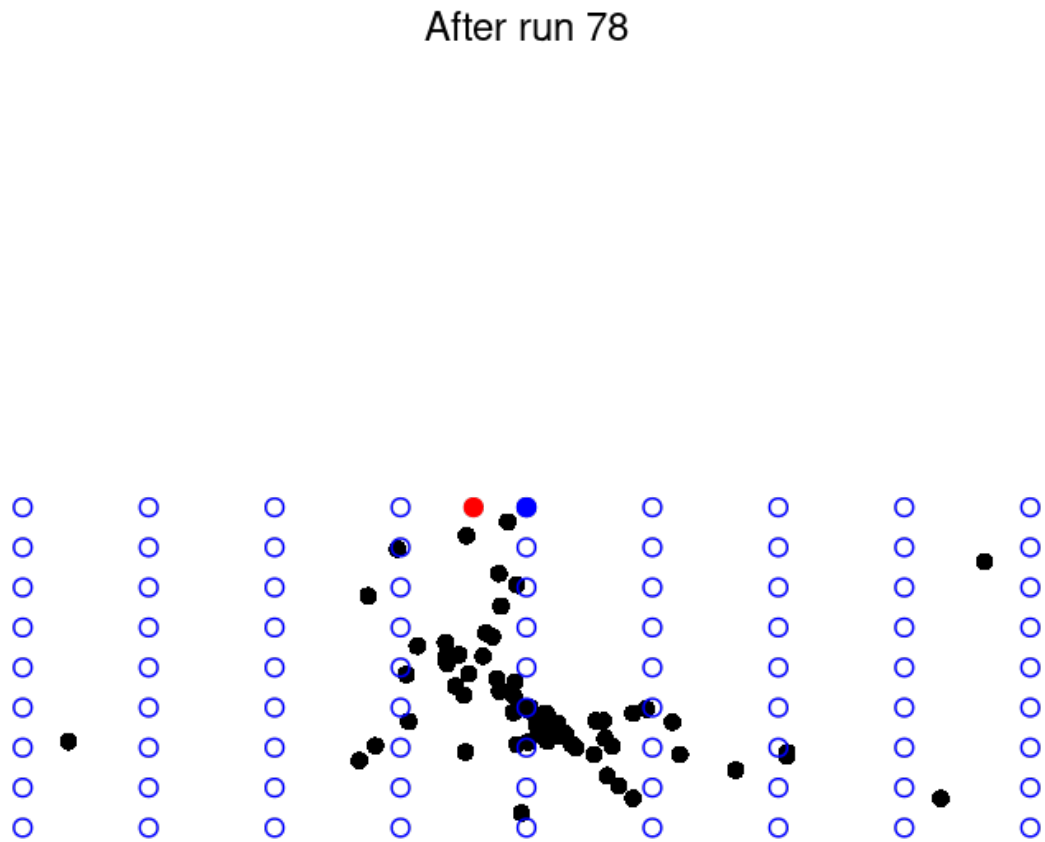
Simple adaptive search:



▸ More details

Bayesian optimization (cont)

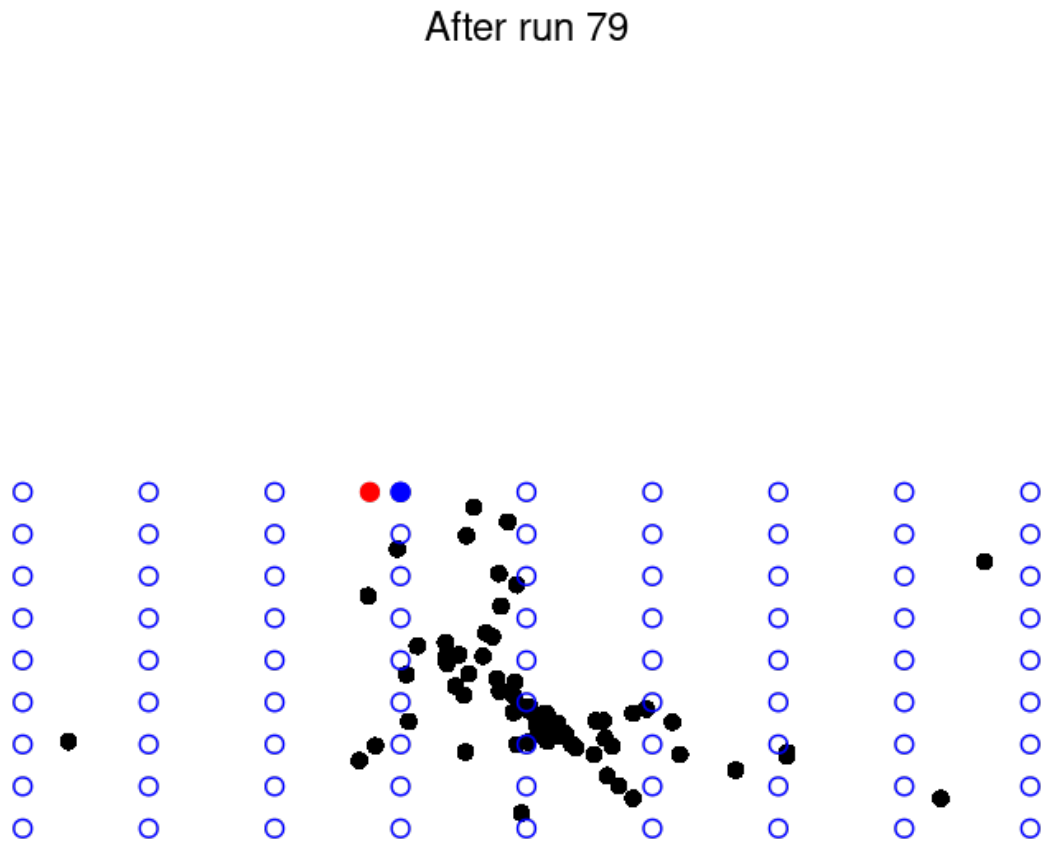
Simple adaptive search:



▸ More details

Bayesian optimization (cont)

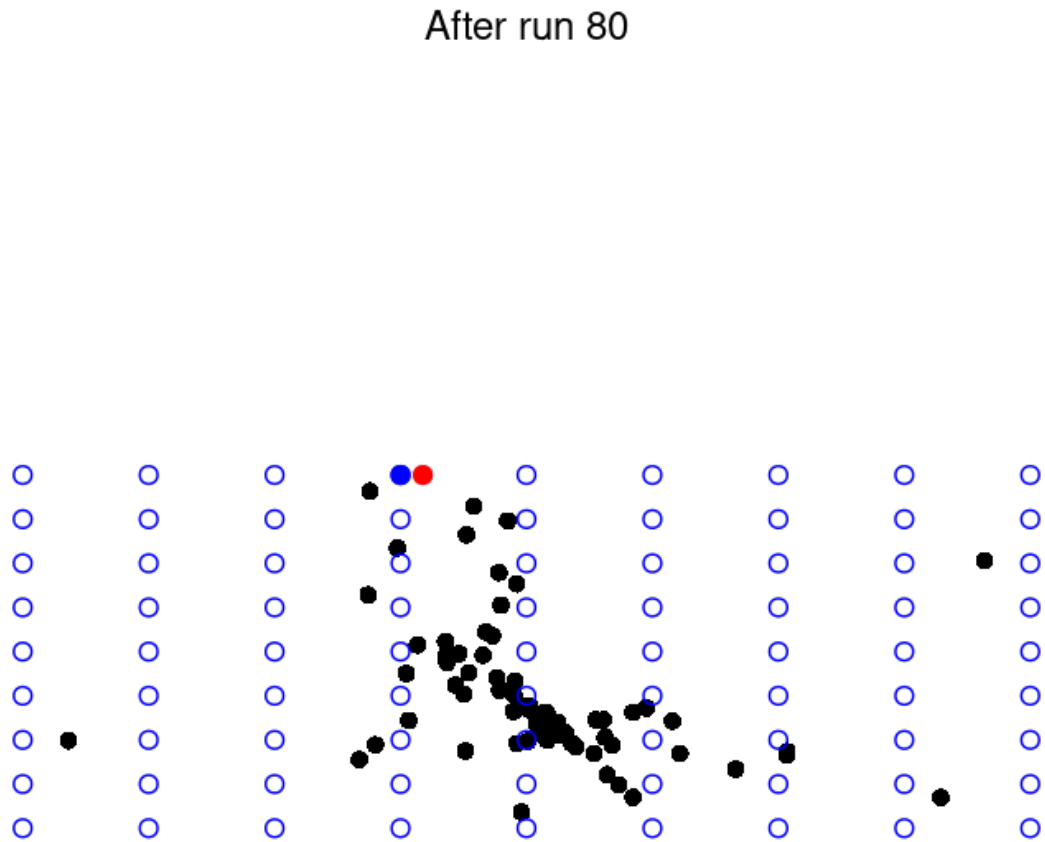
Simple adaptive search:



▸ More details

Bayesian optimization (cont)

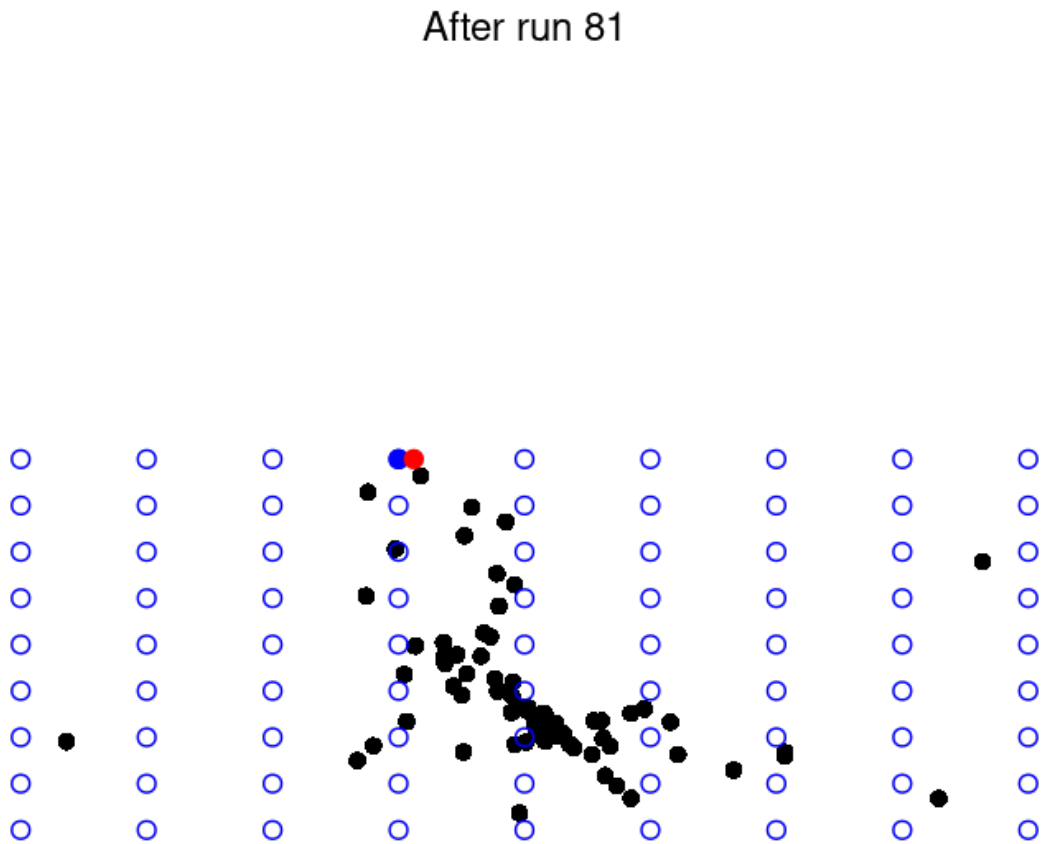
Simple adaptive search:



▸ More details

Bayesian optimization (cont)

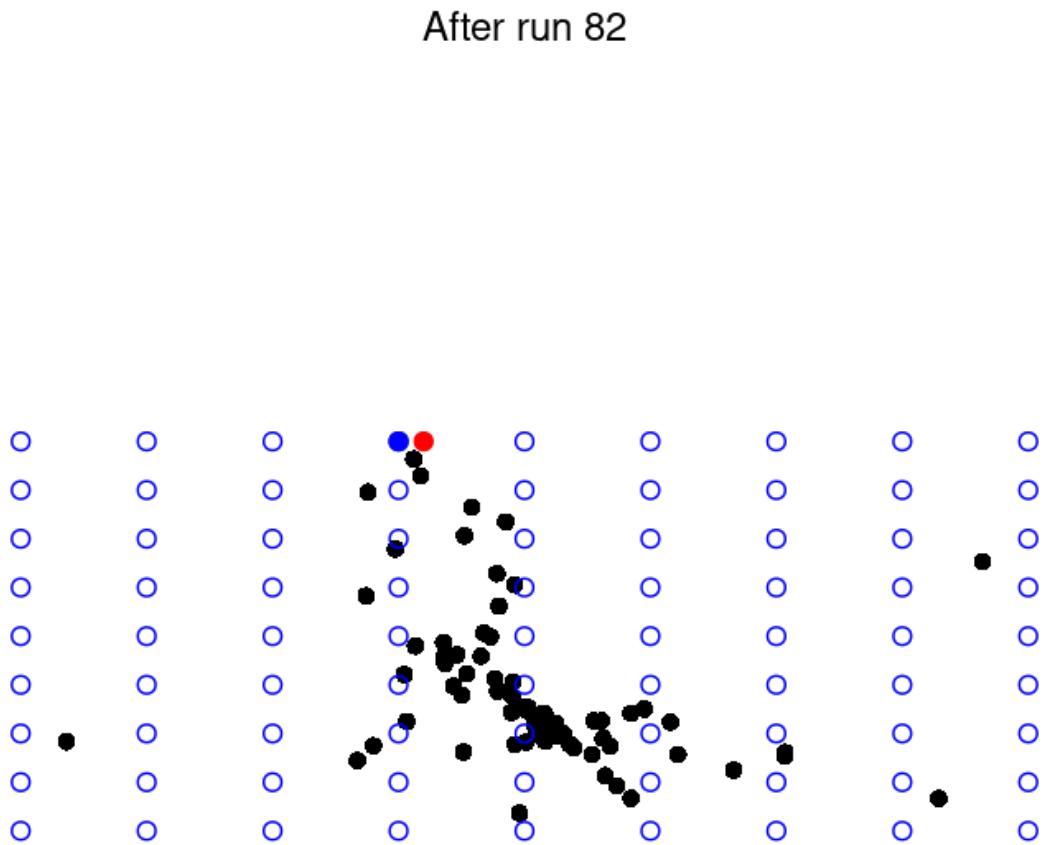
Simple adaptive search:



▸ More details

Bayesian optimization (cont)

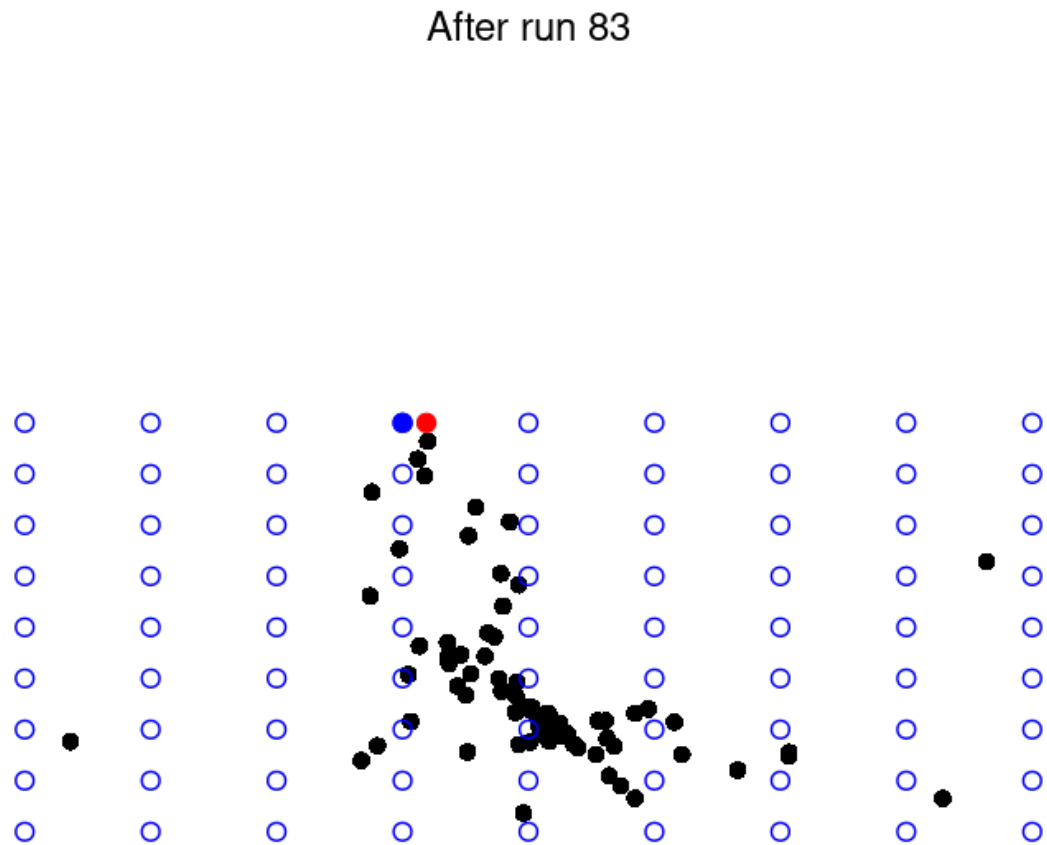
Simple adaptive search:



▸ More details

Bayesian optimization (cont)

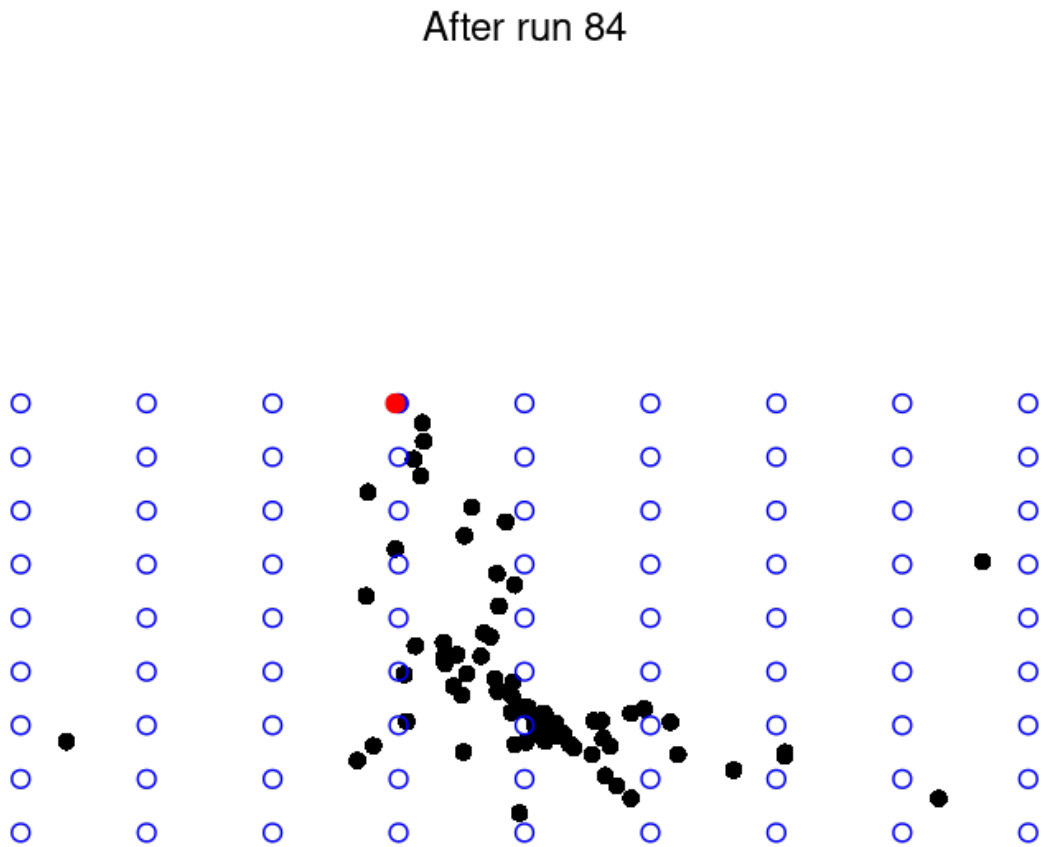
Simple adaptive search:



▸ More details

Bayesian optimization (cont)

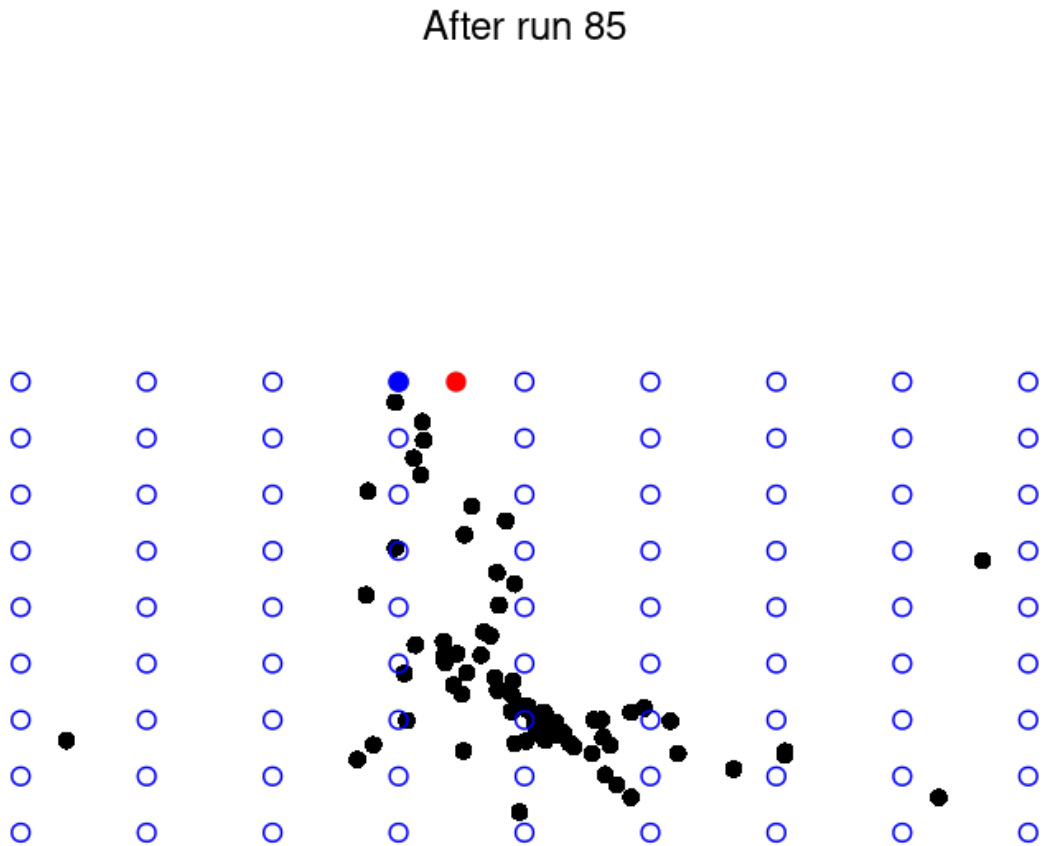
Simple adaptive search:



▸ More details

Bayesian optimization (cont)

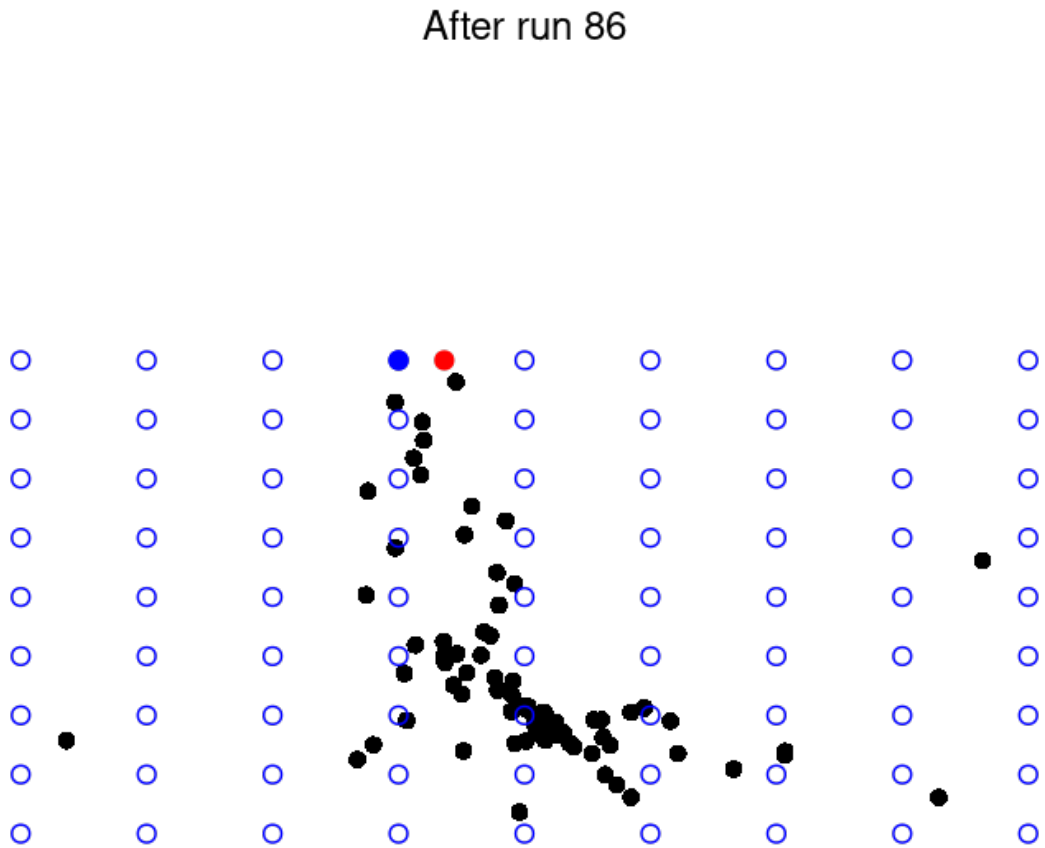
Simple adaptive search:



▸ More details

Bayesian optimization (cont)

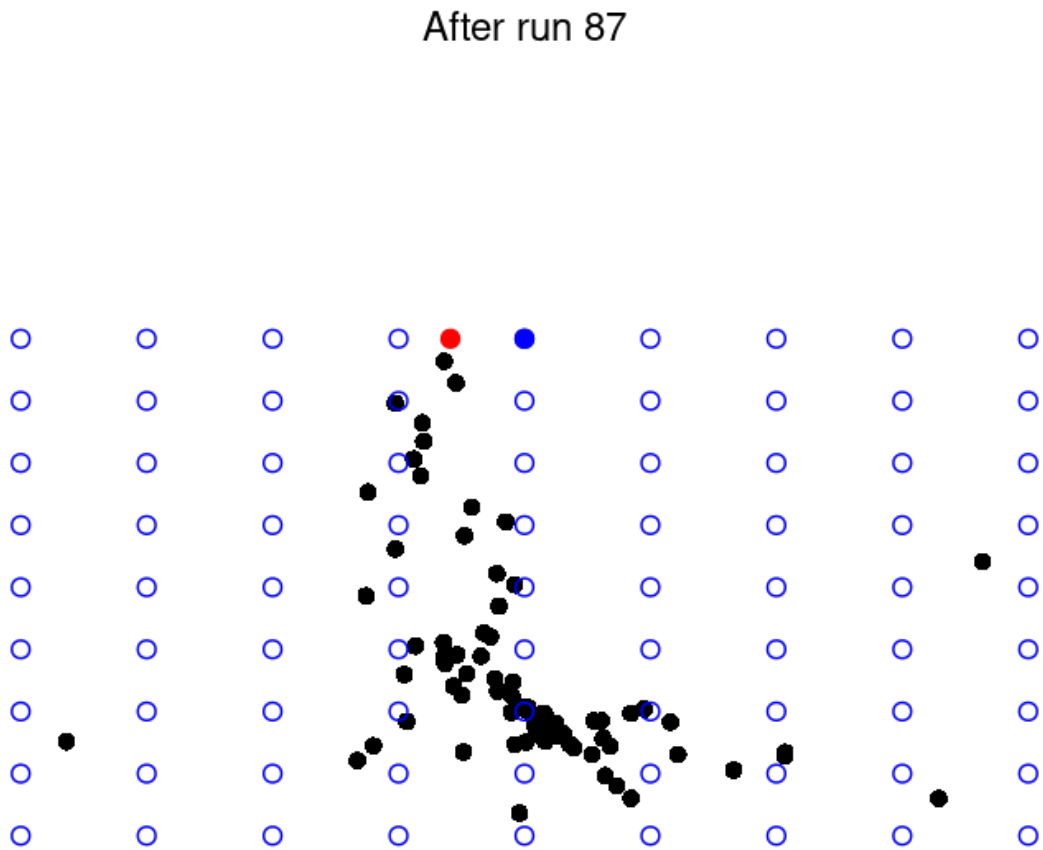
Simple adaptive search:



▸ More details

Bayesian optimization (cont)

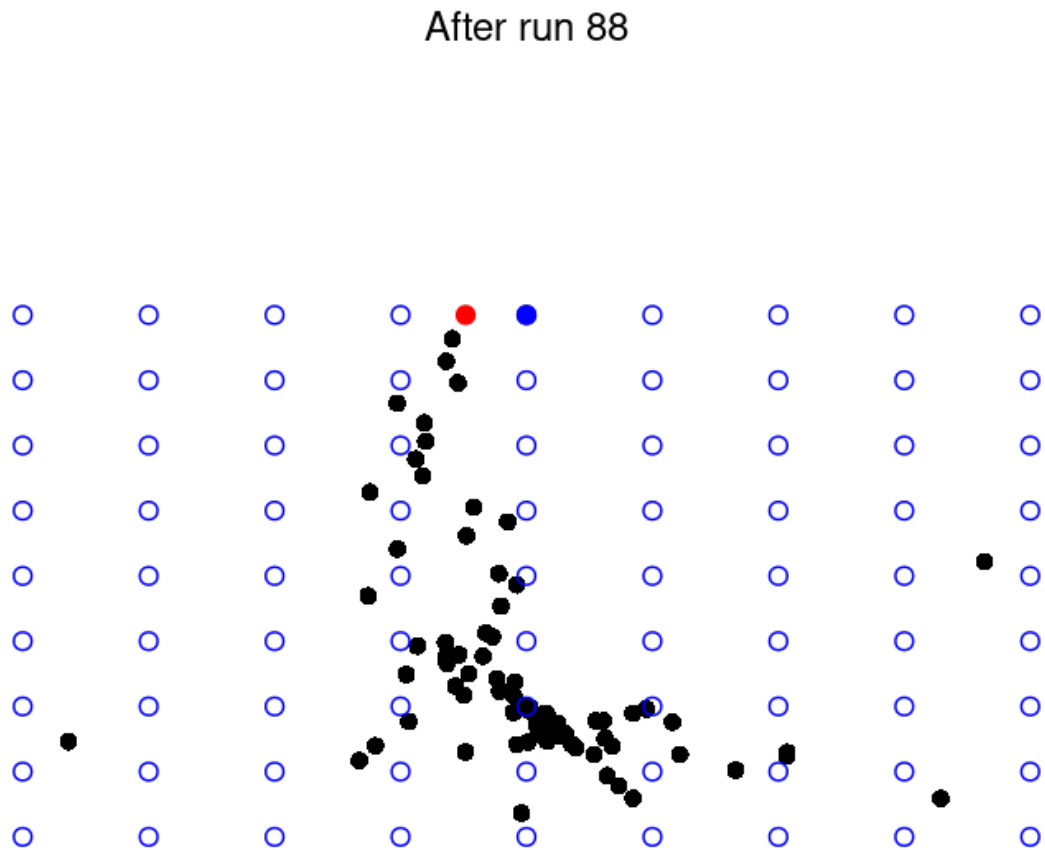
Simple adaptive search:



▸ More details

Bayesian optimization (cont)

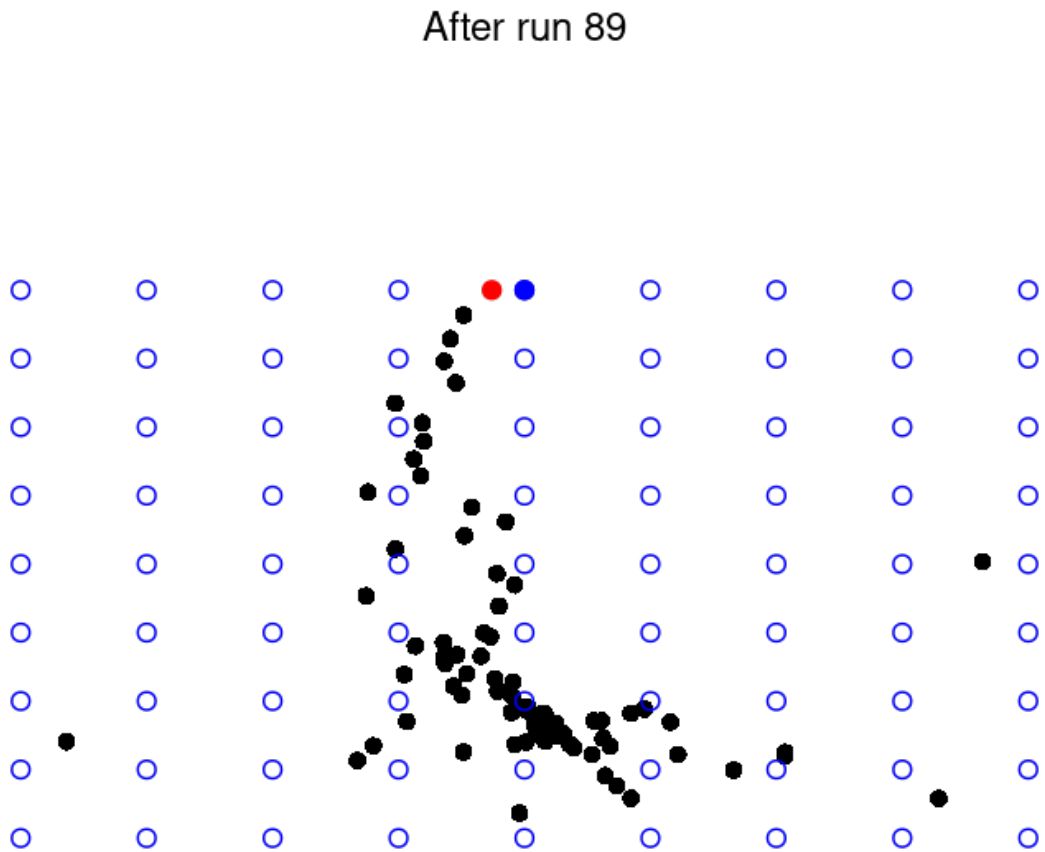
Simple adaptive search:



▸ More details

Bayesian optimization (cont)

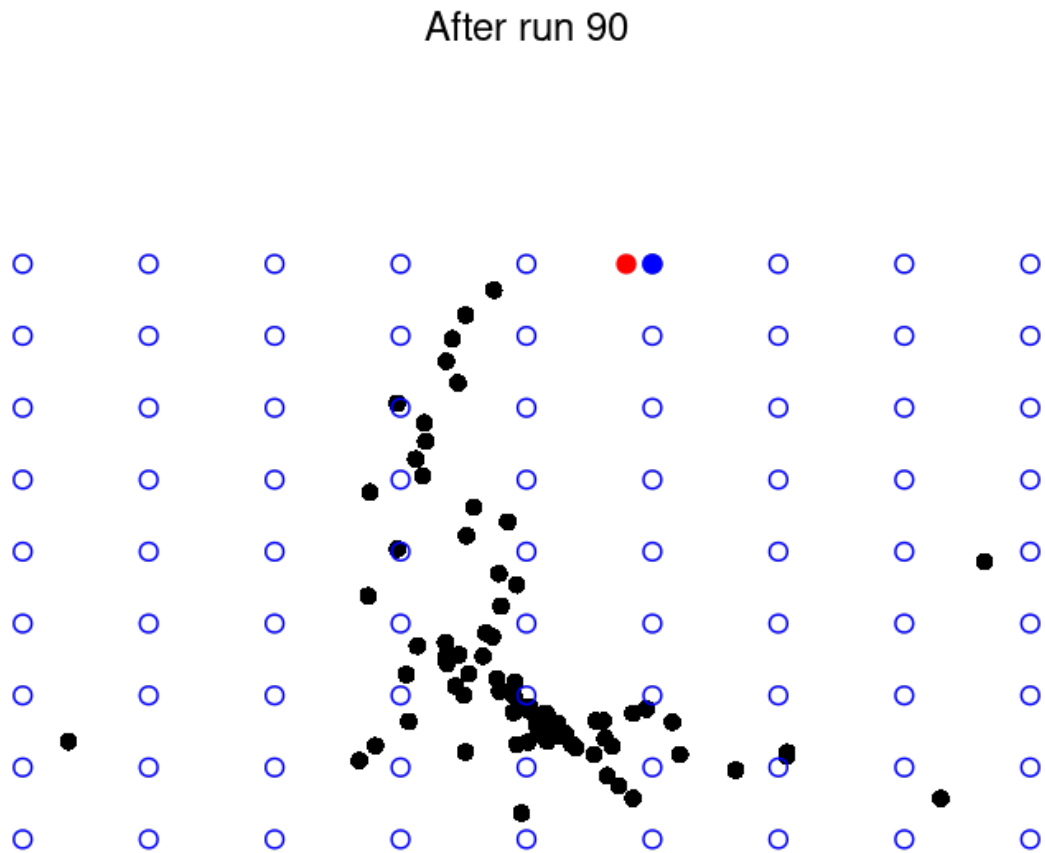
Simple adaptive search:



▸ More details

Bayesian optimization (cont)

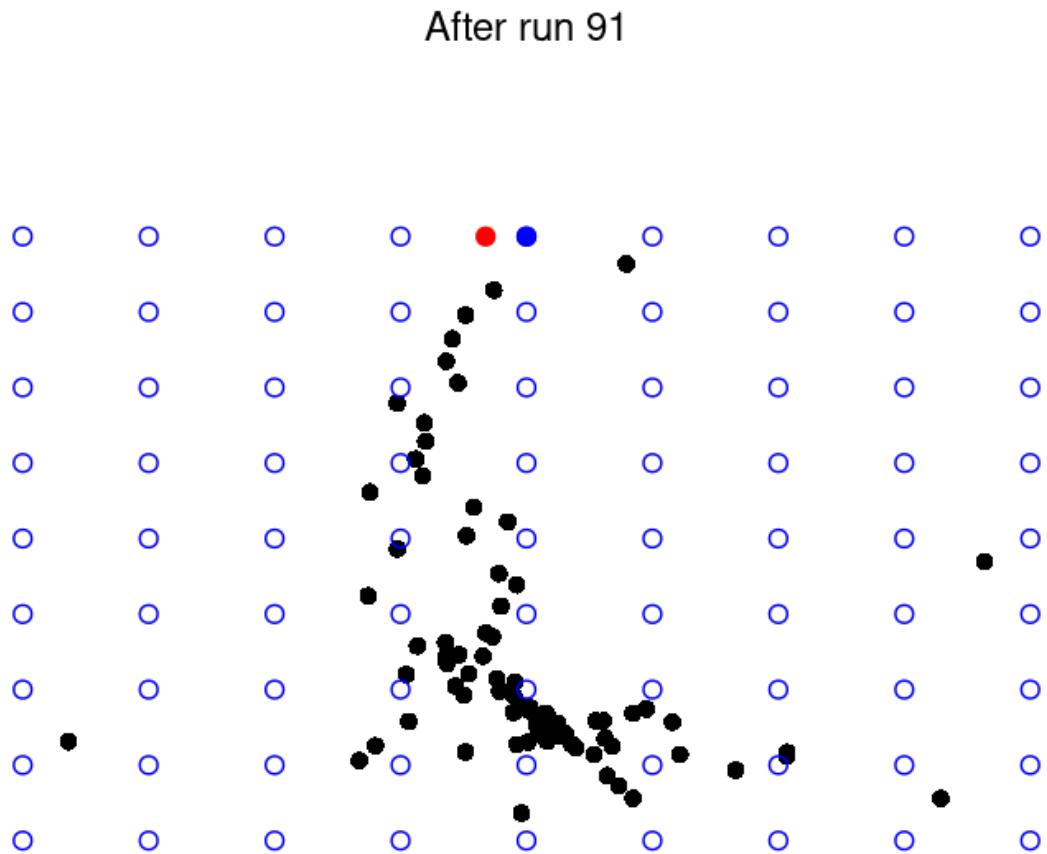
Simple adaptive search:



▸ More details

Bayesian optimization (cont)

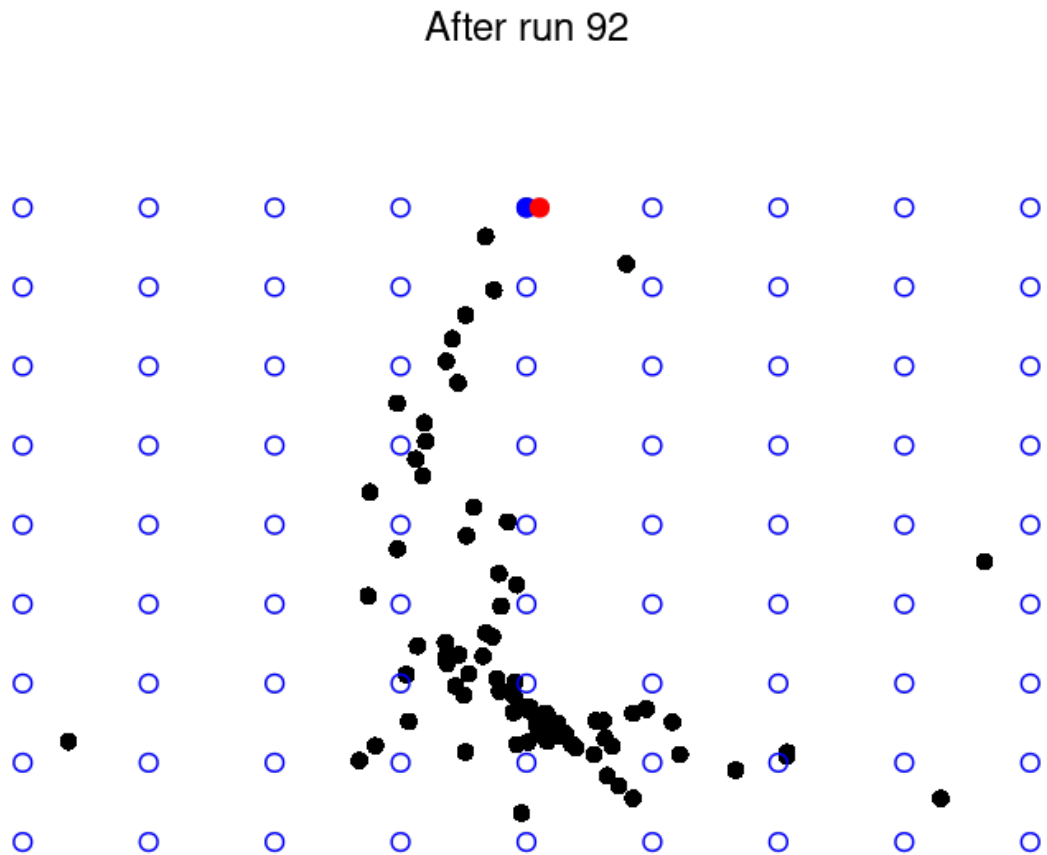
Simple adaptive search:



▸ More details

Bayesian optimization (cont)

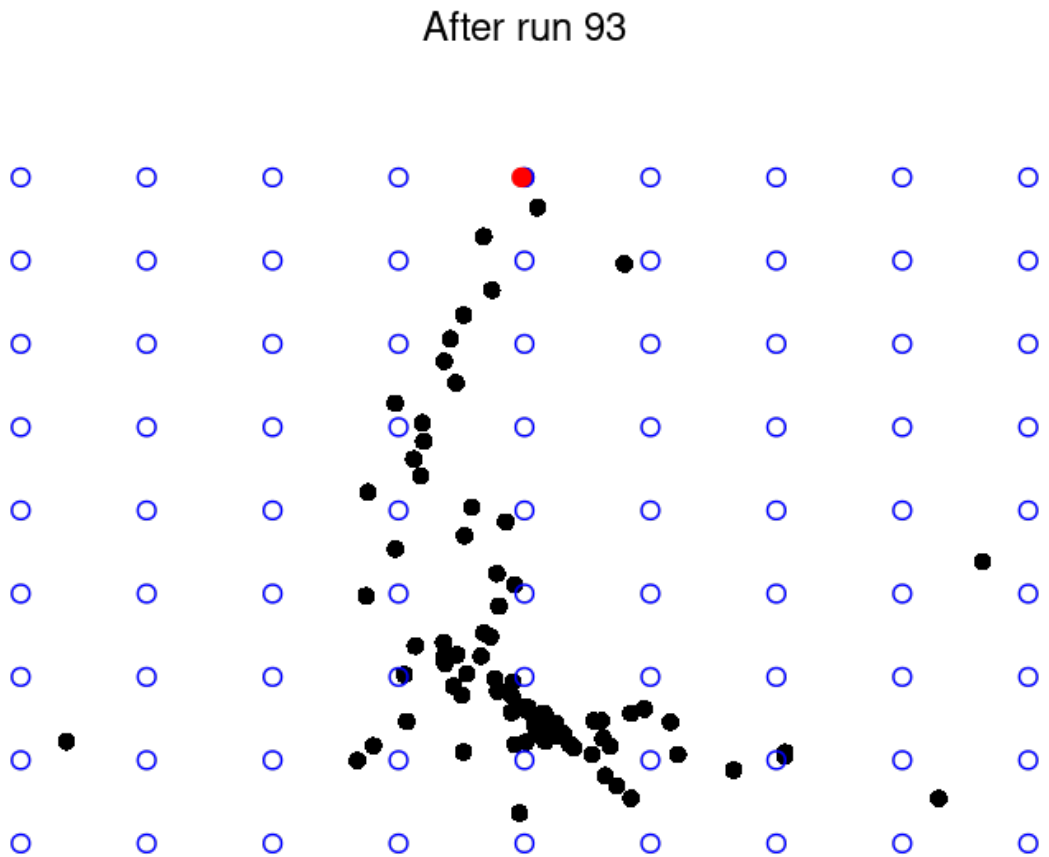
Simple adaptive search:



▸ More details

Bayesian optimization (cont)

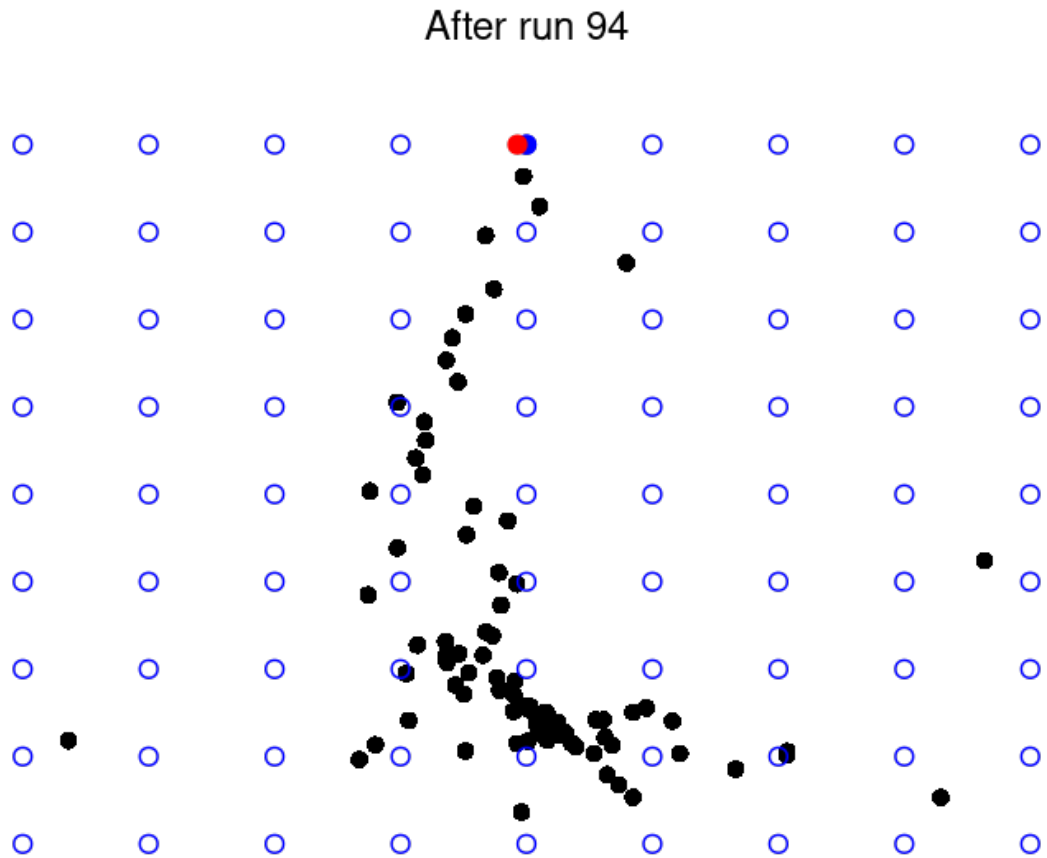
Simple adaptive search:



▸ More details

Bayesian optimization (cont)

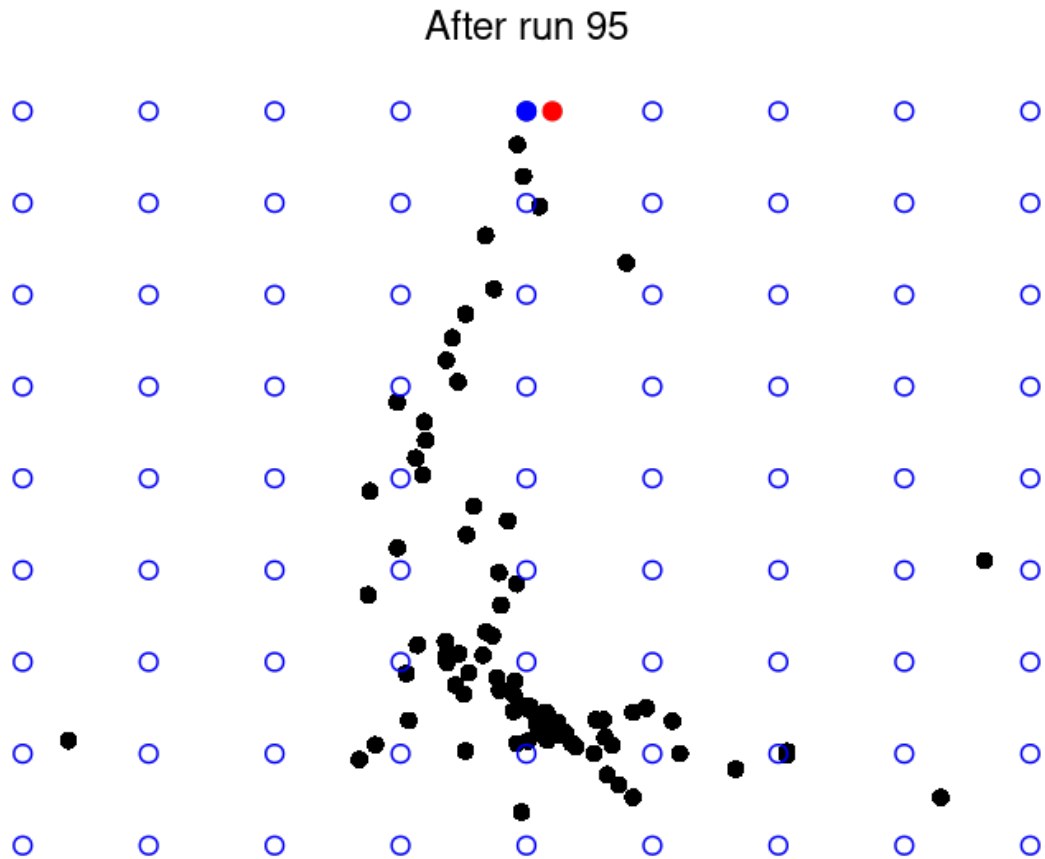
Simple adaptive search:



▸ More details

Bayesian optimization (cont)

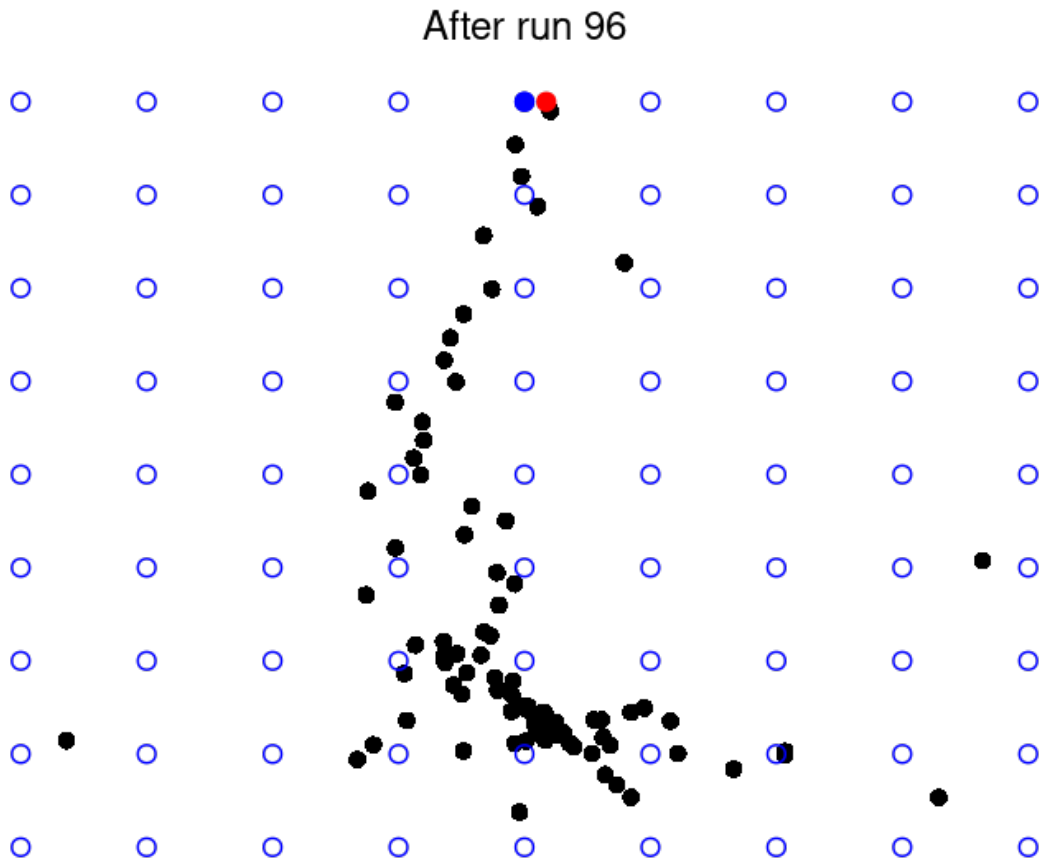
Simple adaptive search:



▸ More details

Bayesian optimization (cont)

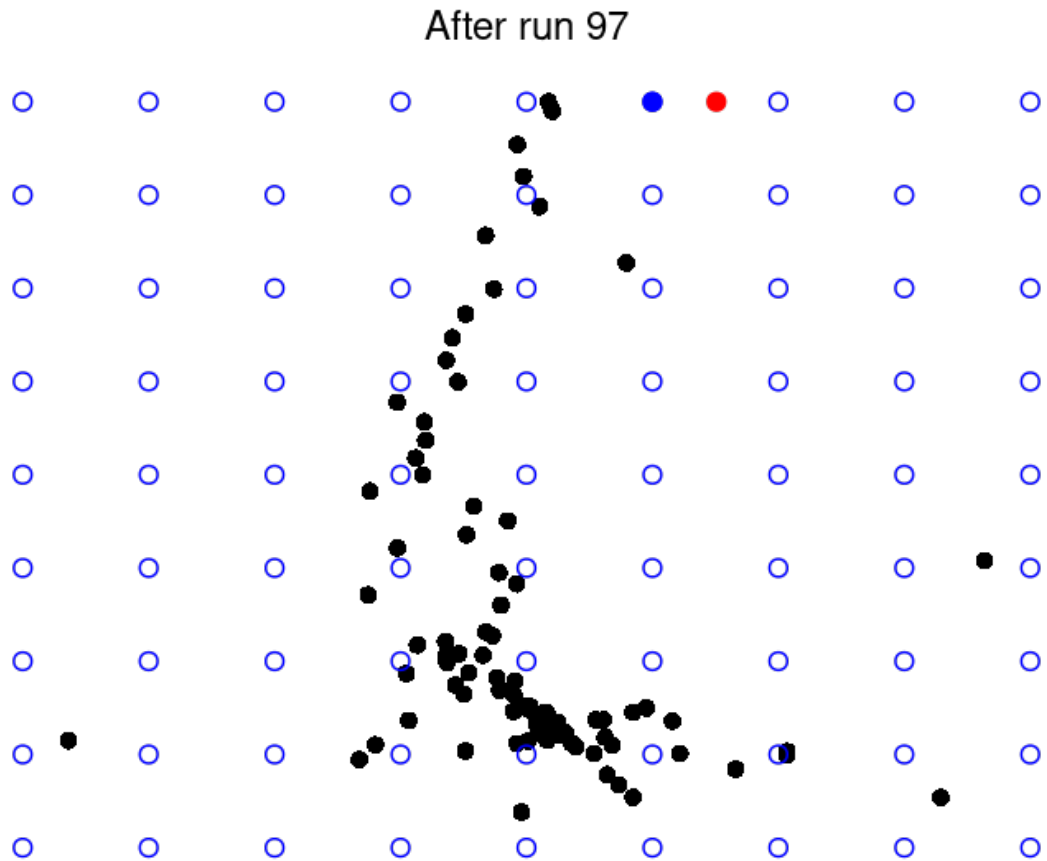
Simple adaptive search:



▸ More details

Bayesian optimization (cont)

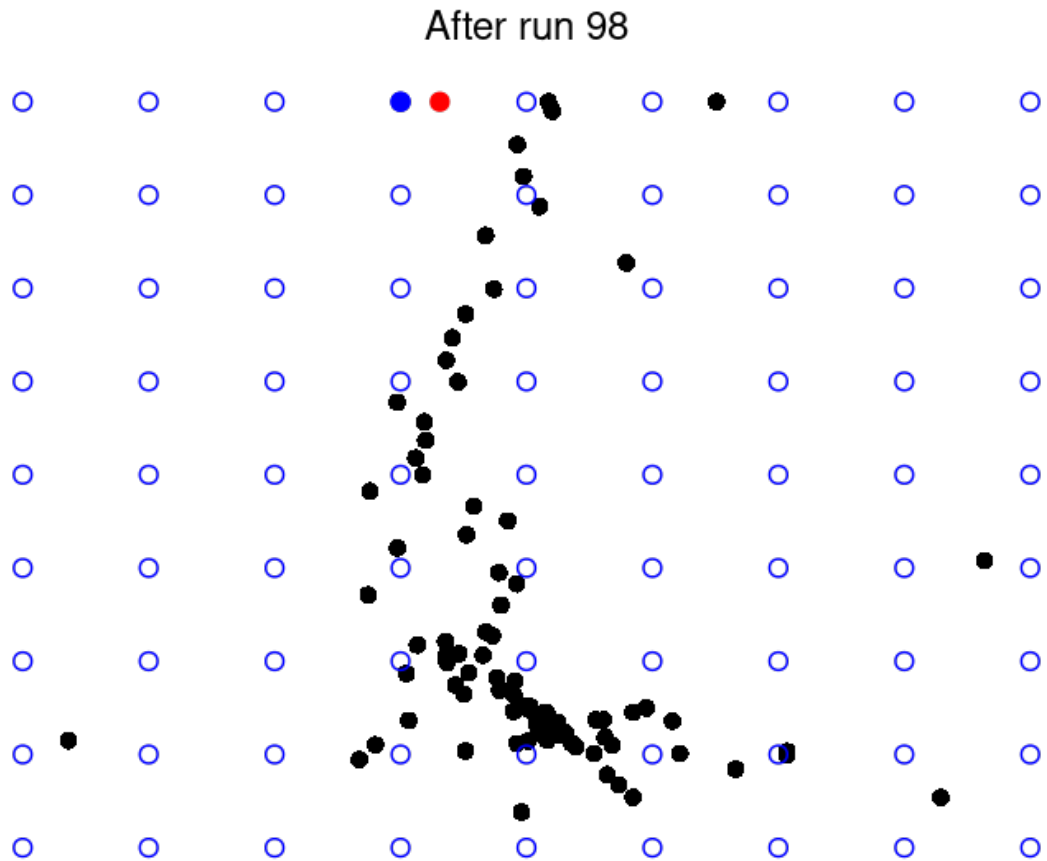
Simple adaptive search:



▸ More details

Bayesian optimization (cont)

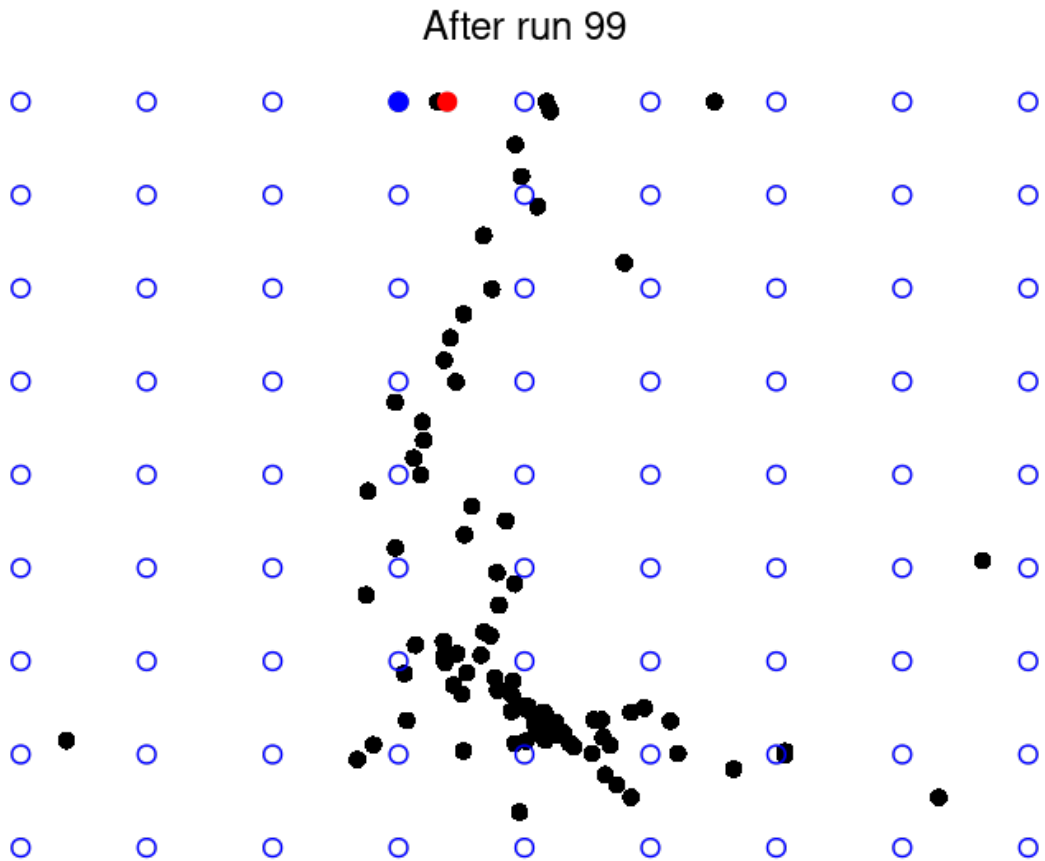
Simple adaptive search:



▸ More details

Bayesian optimization (cont)

Simple adaptive search:



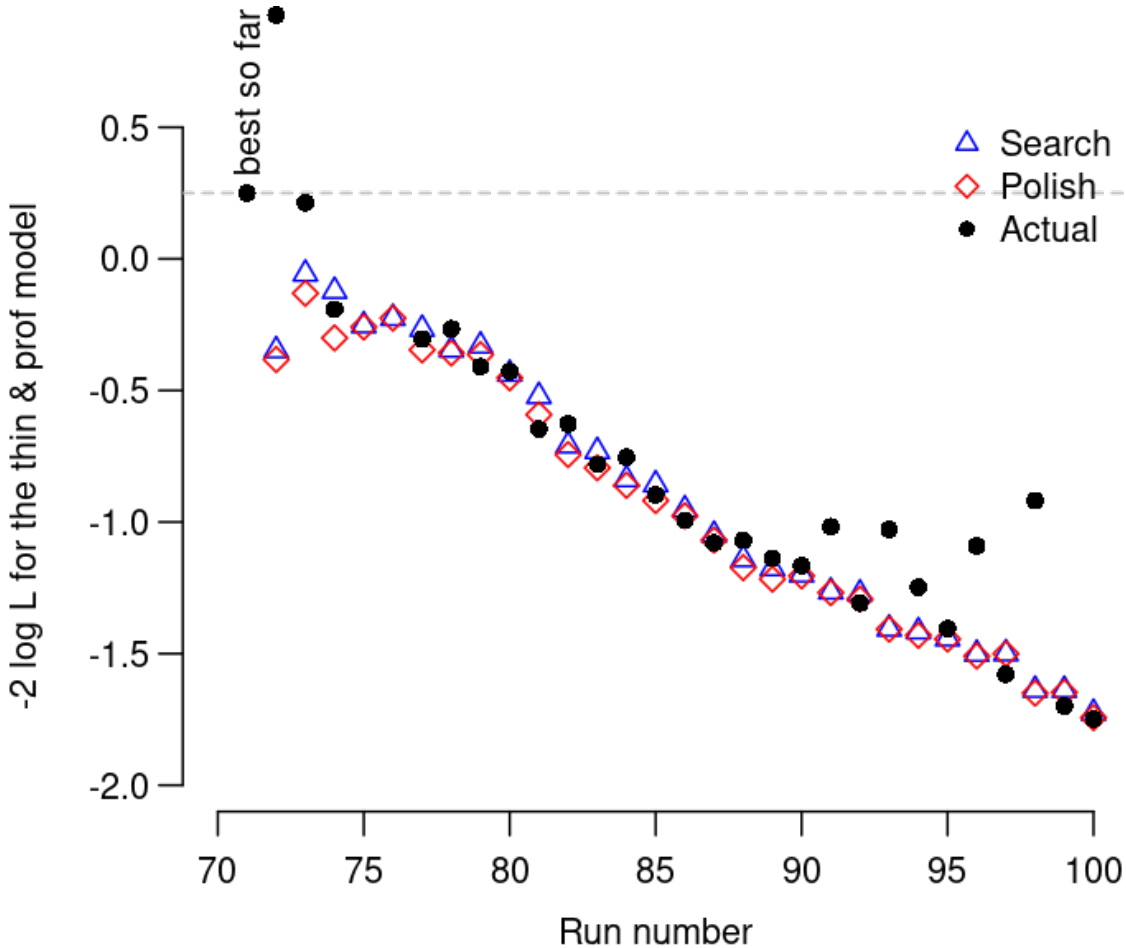
▸ More details

Results

I use the `RobustGaSP` emulator (Gu et al., 2018) with linear and quadratic trend (centred), a 9^6-point grid for $\mathcal{S}^+$ (half a million points), and `optim(method = "L-BFGS-B")` for the quasi-Newton method.

Results

I use the `RobustGaSP` emulator (Gu et al., 2018) with linear and quadratic trend (centred), a 9^6-point grid for $\mathcal{S}^+$ (half a million points), and `optim(method = "L-BFGS-B")` for the quasi-Newton method.

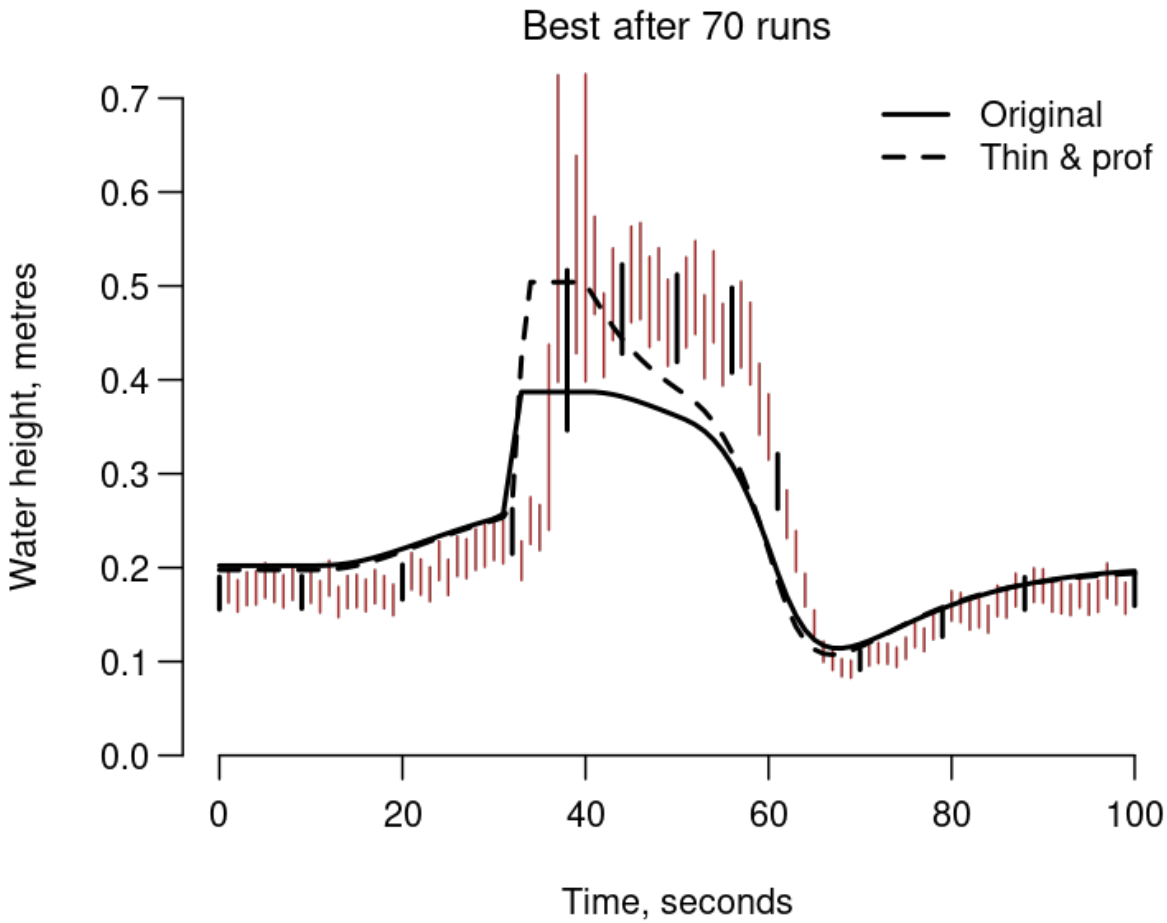


Second phase of runs

Original continues with Downhill Simplex. Thin & prof switches to Bayesian optimization.

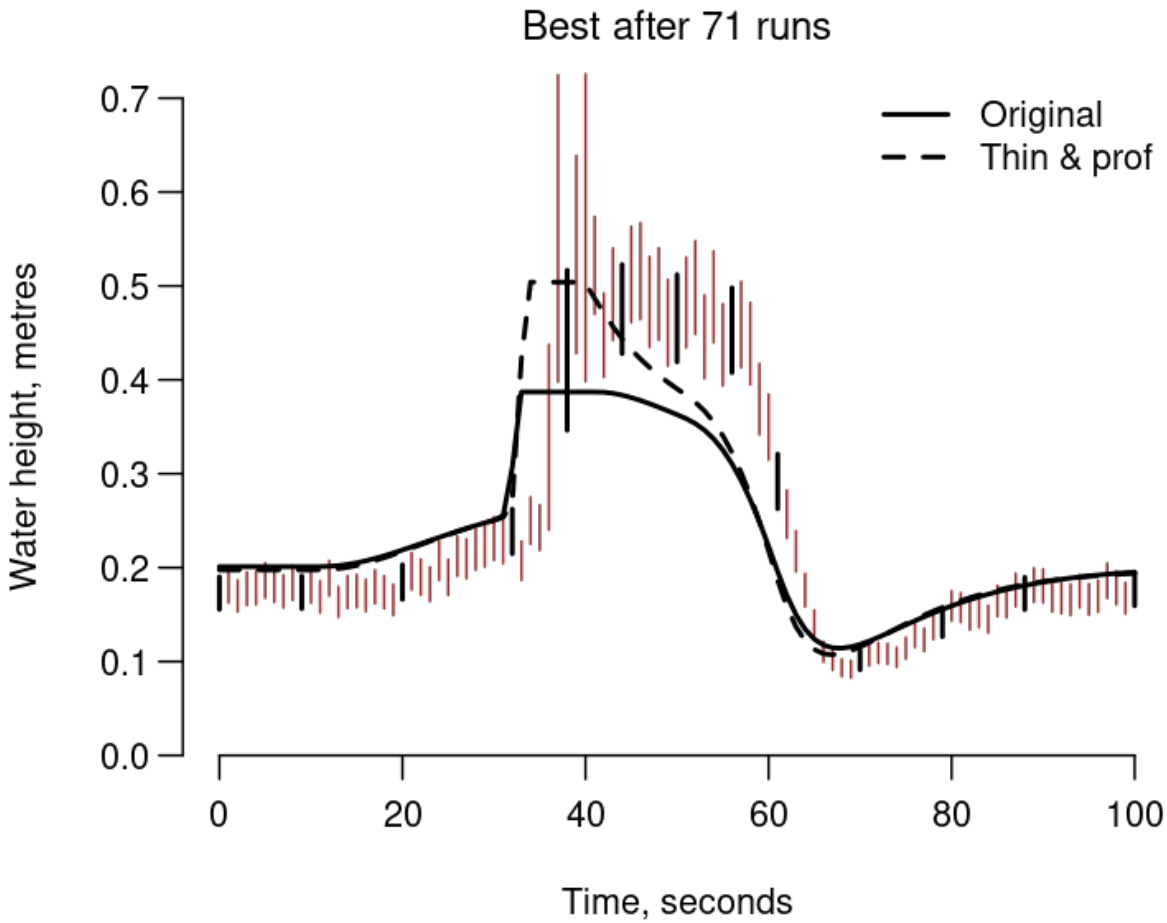
Second phase of runs

Original continues with Downhill Simplex. Thin & prof switches to Bayesian optimization.



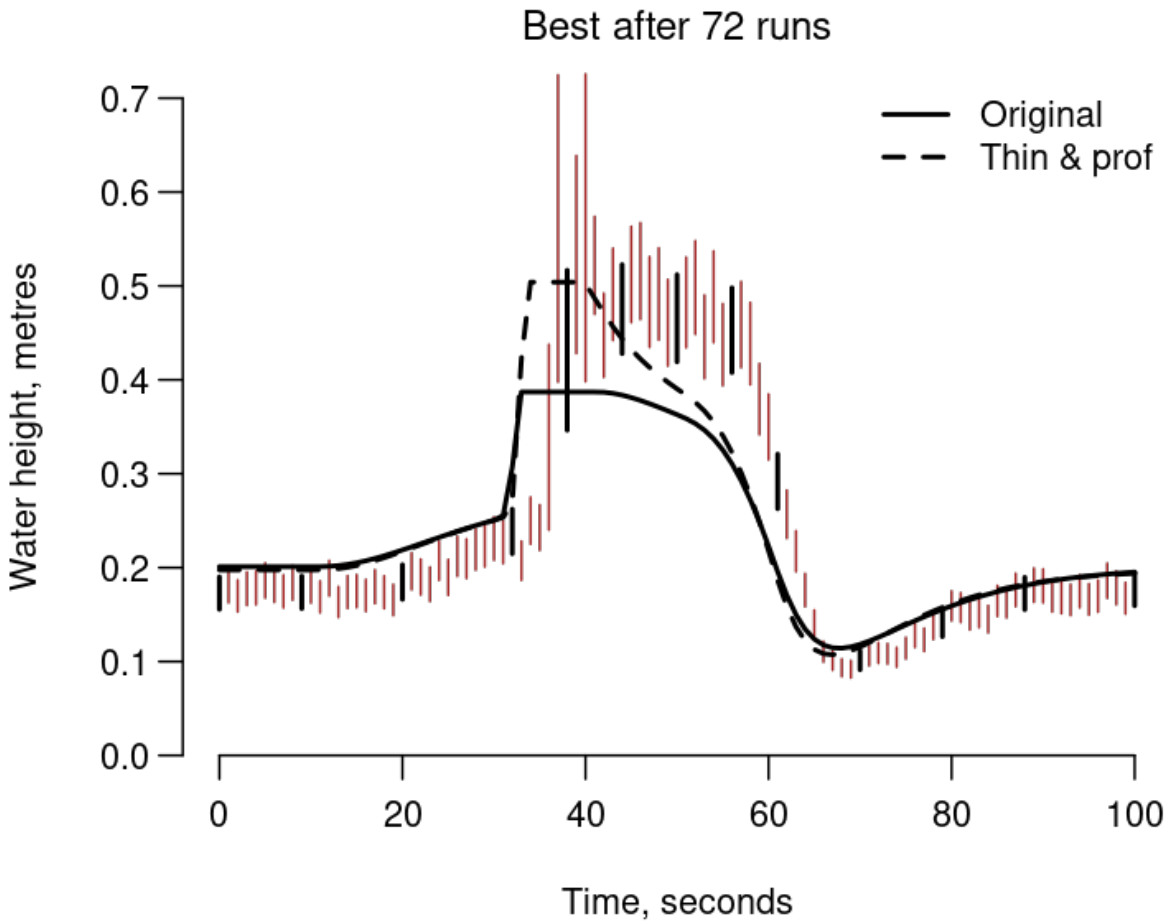
Second phase of runs

Original continues with Downhill Simplex. Thin & prof switches to Bayesian optimization.



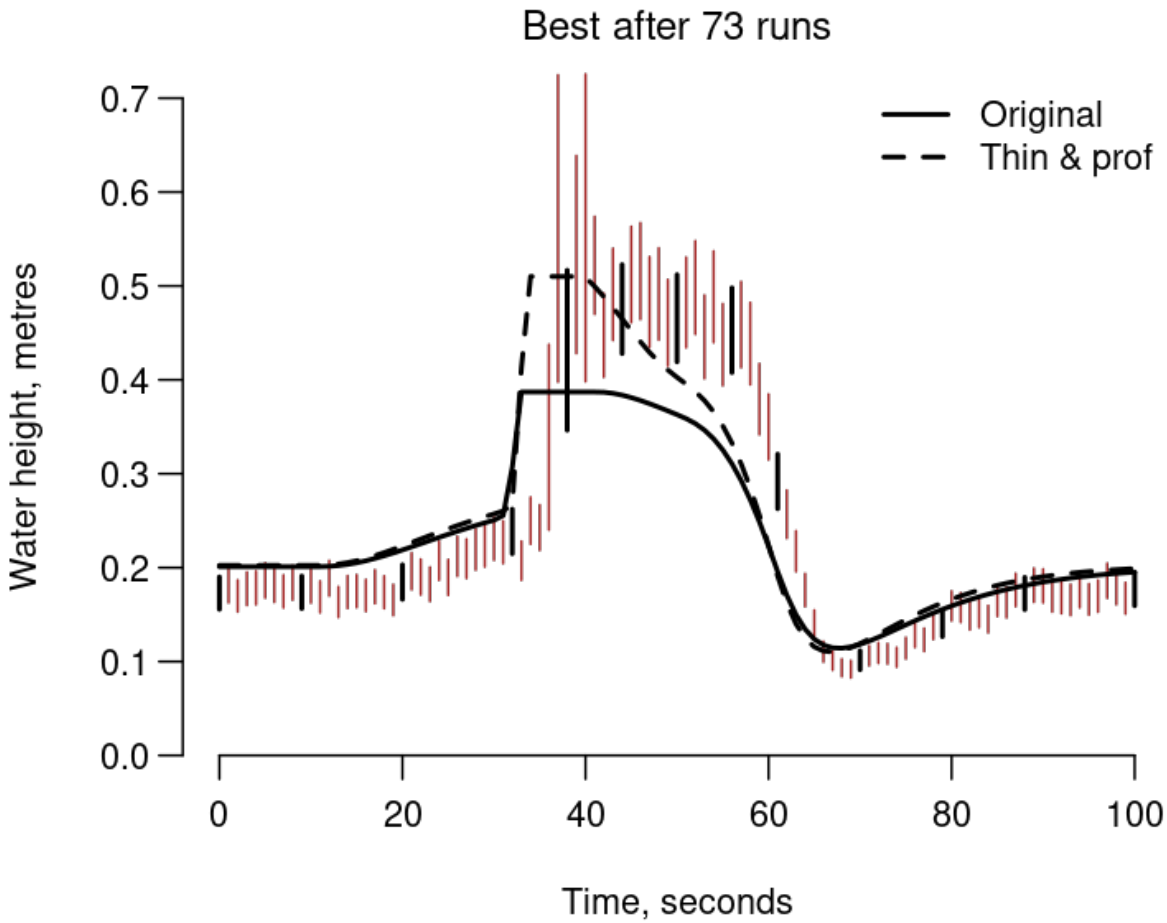
Second phase of runs

Original continues with Downhill Simplex. Thin & prof switches to Bayesian optimization.



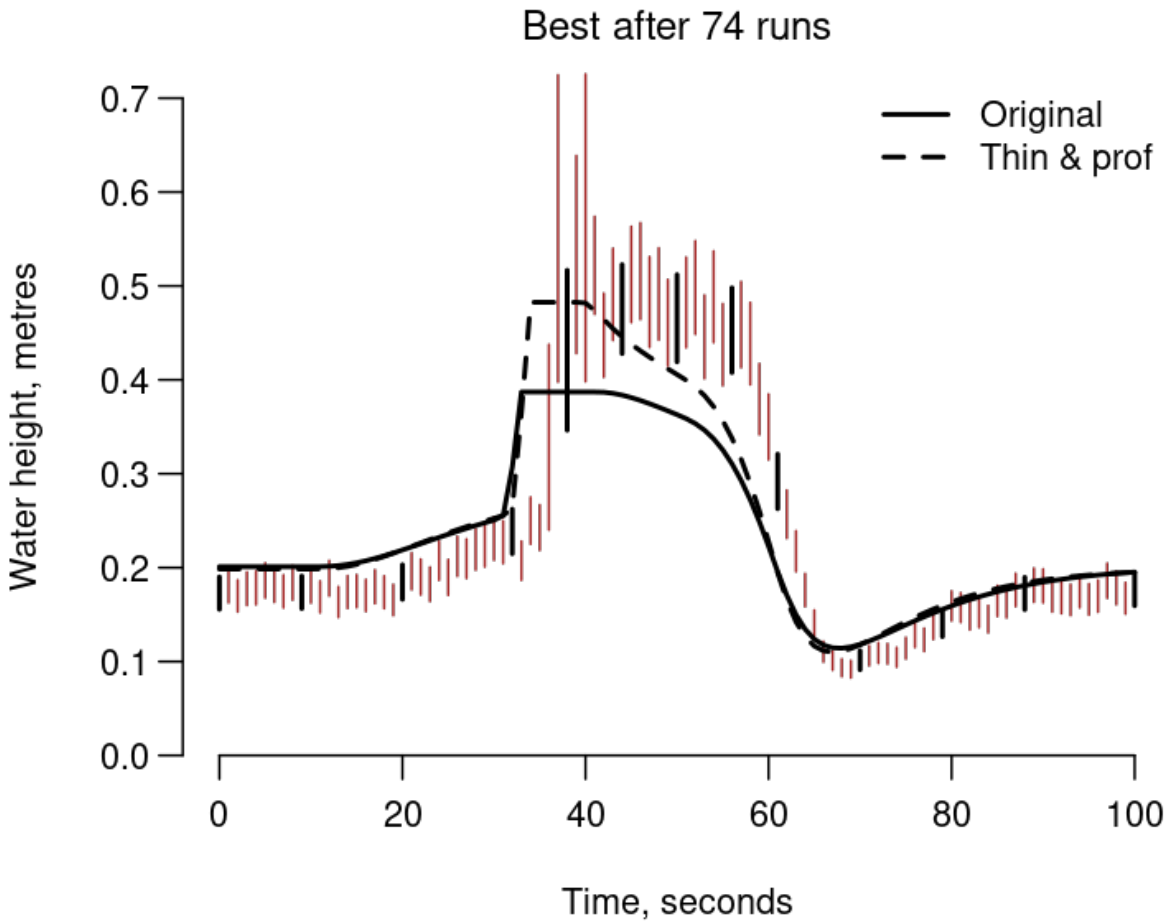
Second phase of runs

Original continues with Downhill Simplex. Thin & prof switches to Bayesian optimization.



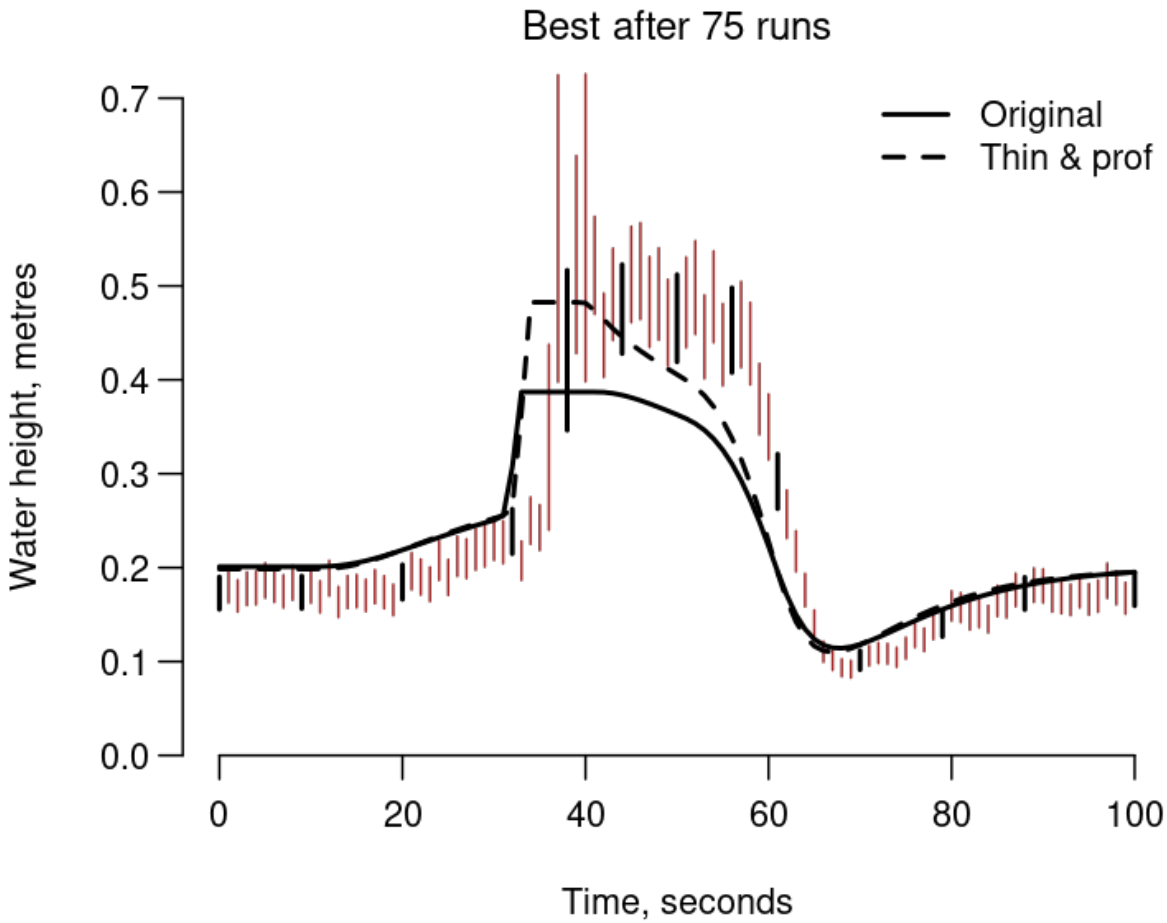
Second phase of runs

Original continues with Downhill Simplex. Thin & prof switches to Bayesian optimization.



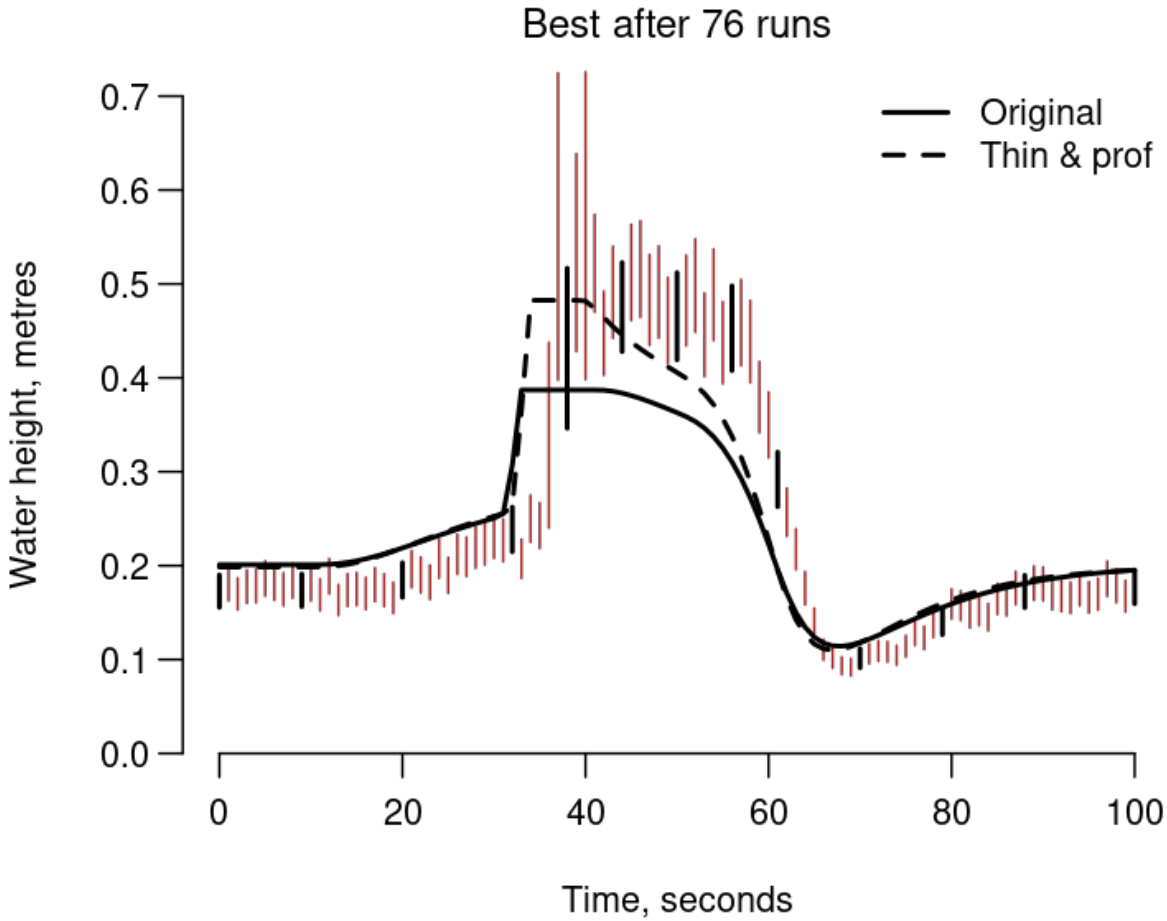
Second phase of runs

Original continues with Downhill Simplex. Thin & prof switches to Bayesian optimization.



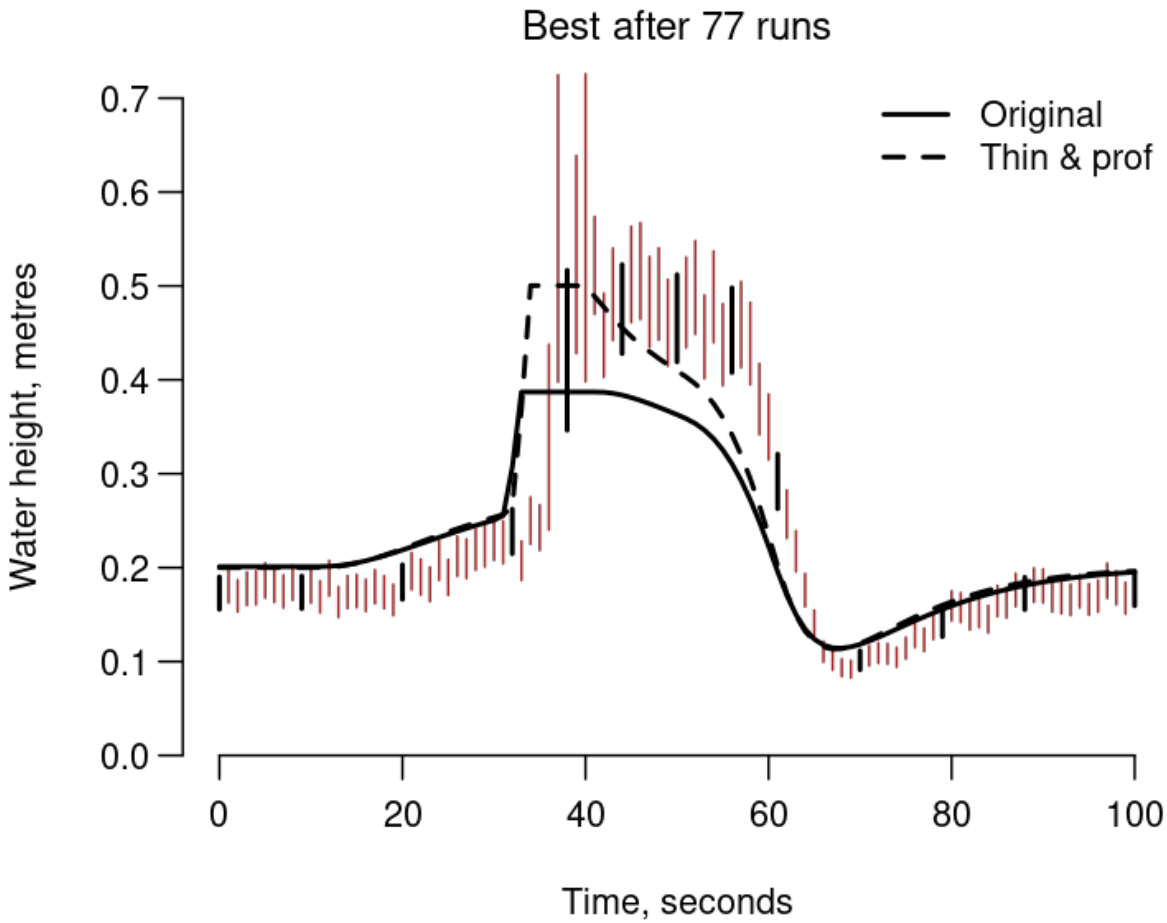
Second phase of runs

Original continues with Downhill Simplex. Thin & prof switches to Bayesian optimization.



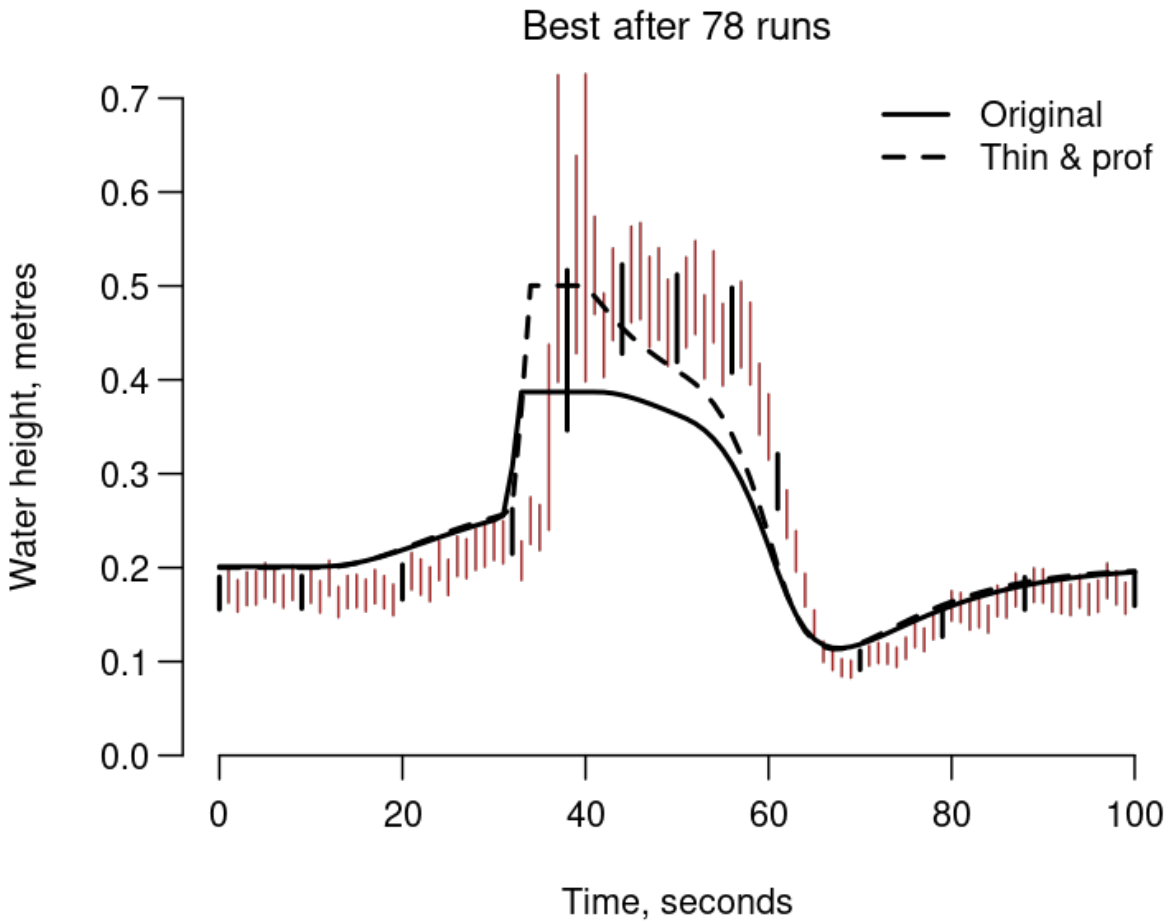
Second phase of runs

Original continues with Downhill Simplex. Thin & prof switches to Bayesian optimization.



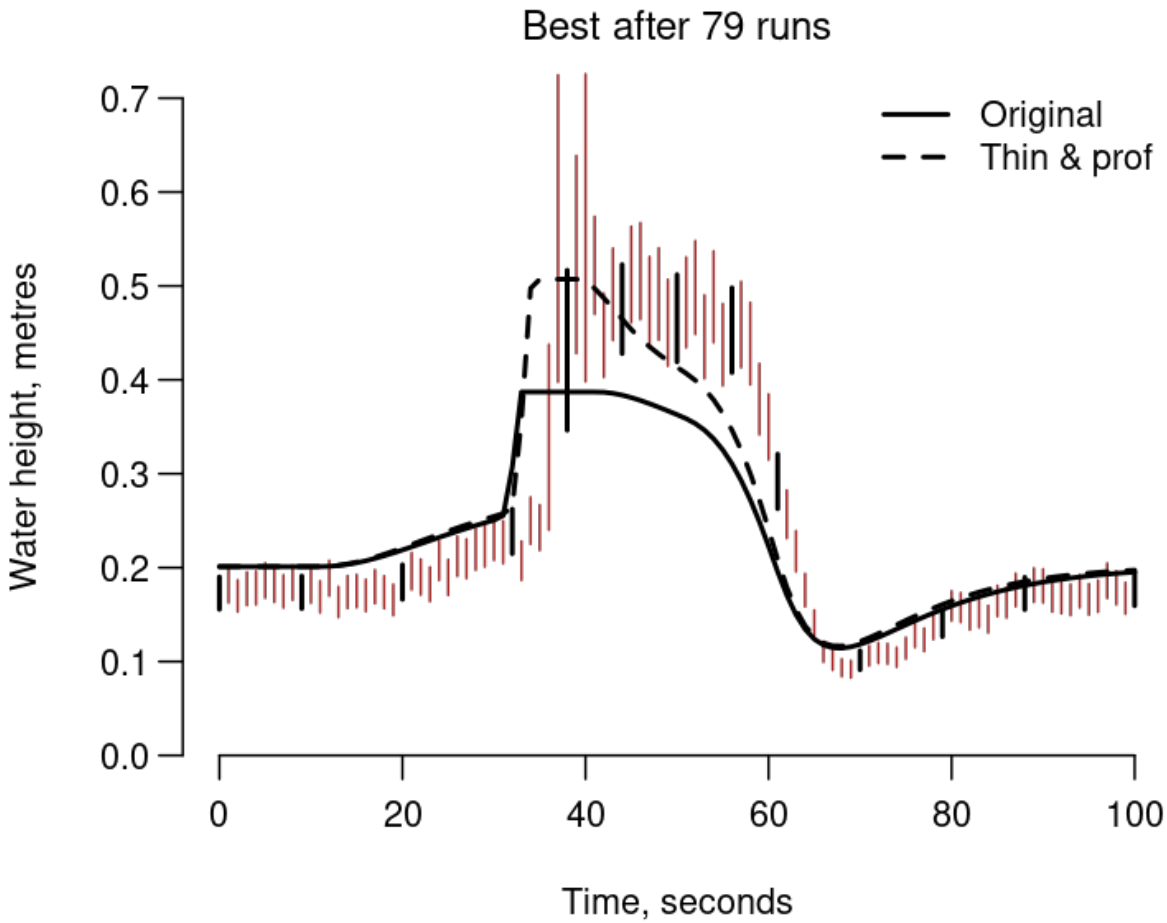
Second phase of runs

Original continues with Downhill Simplex. Thin & prof switches to Bayesian optimization.



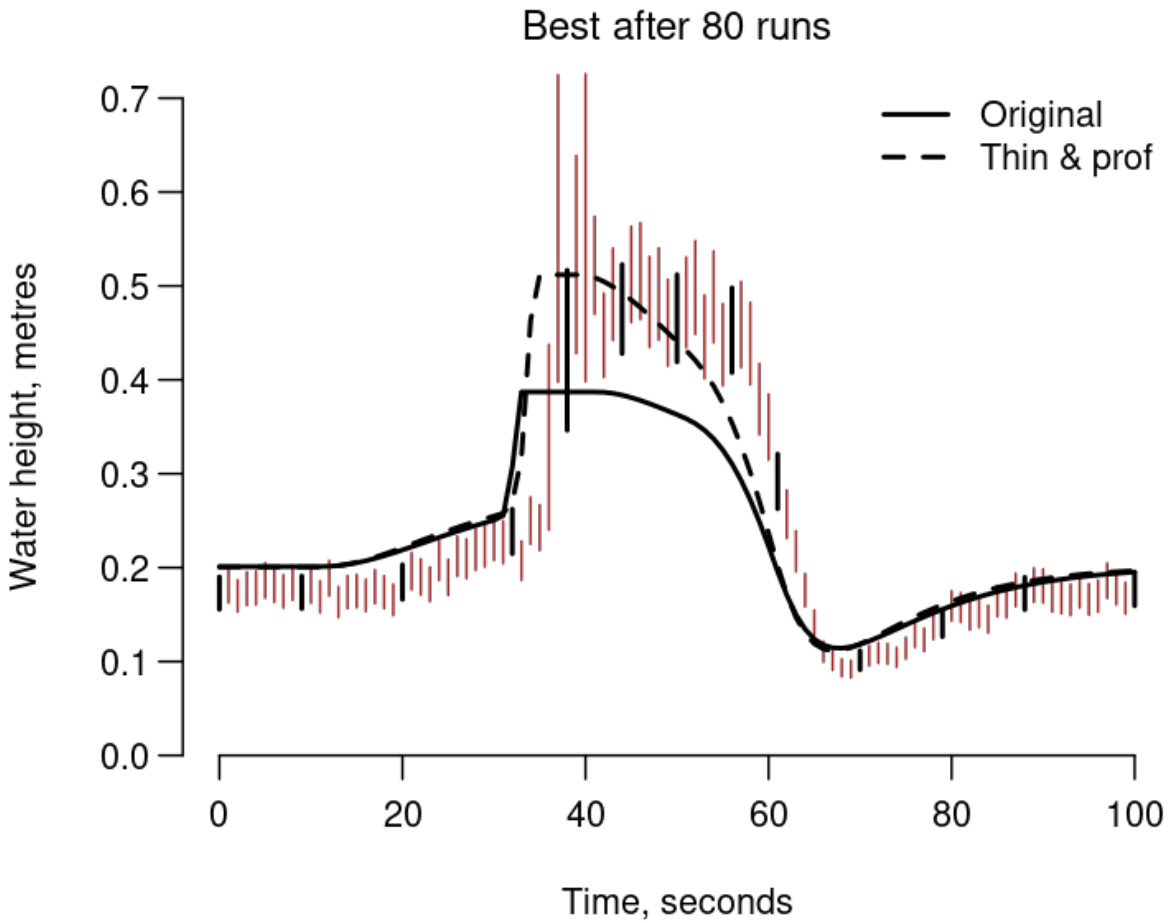
Second phase of runs

Original continues with Downhill Simplex. Thin & prof switches to Bayesian optimization.



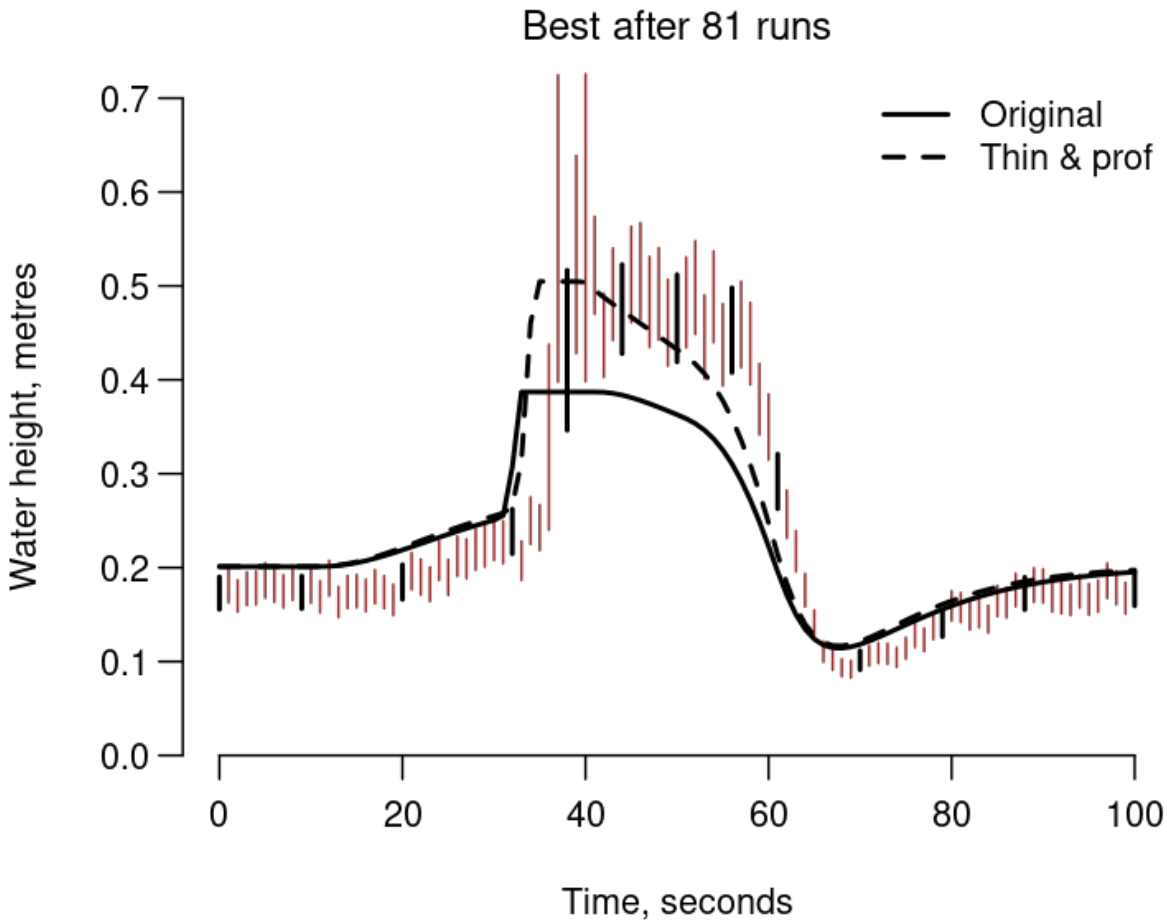
Second phase of runs

Original continues with Downhill Simplex. Thin & prof switches to Bayesian optimization.



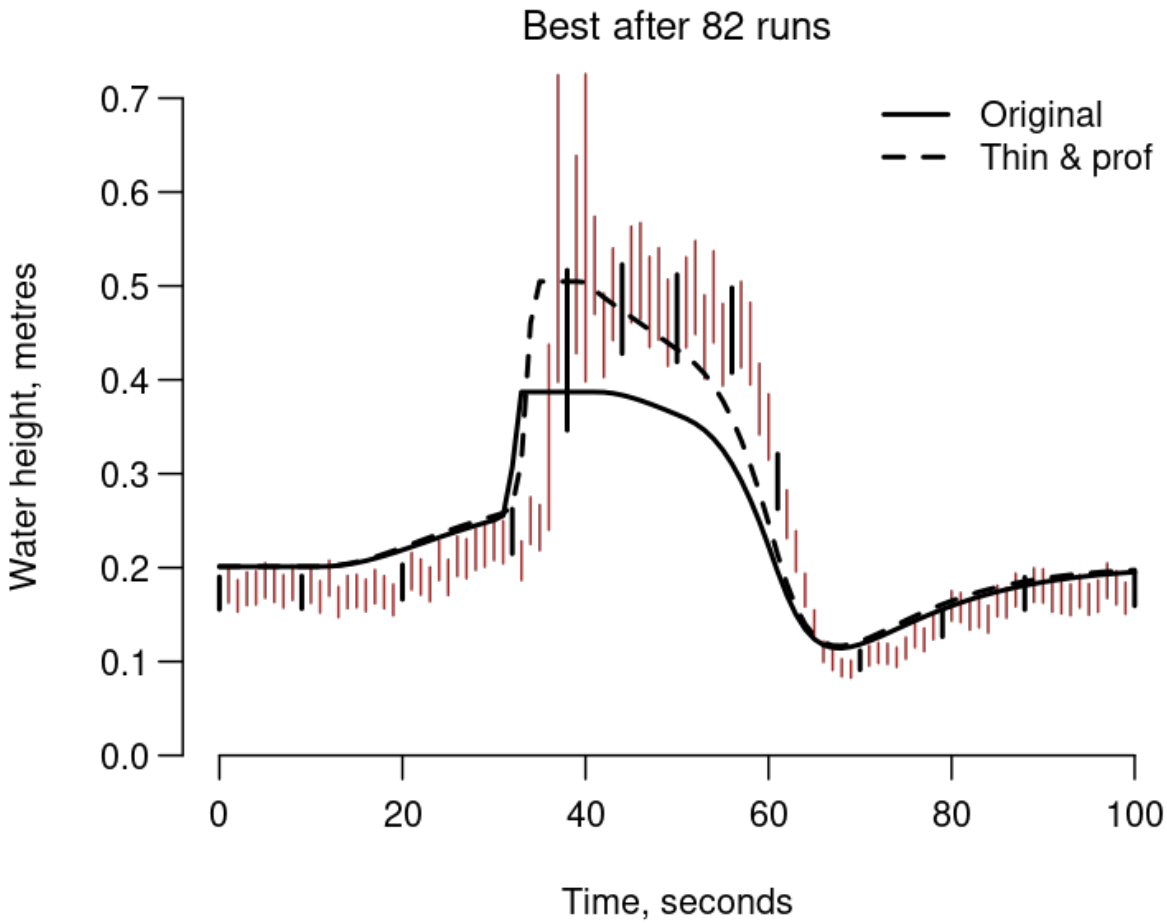
Second phase of runs

Original continues with Downhill Simplex. Thin & prof switches to Bayesian optimization.



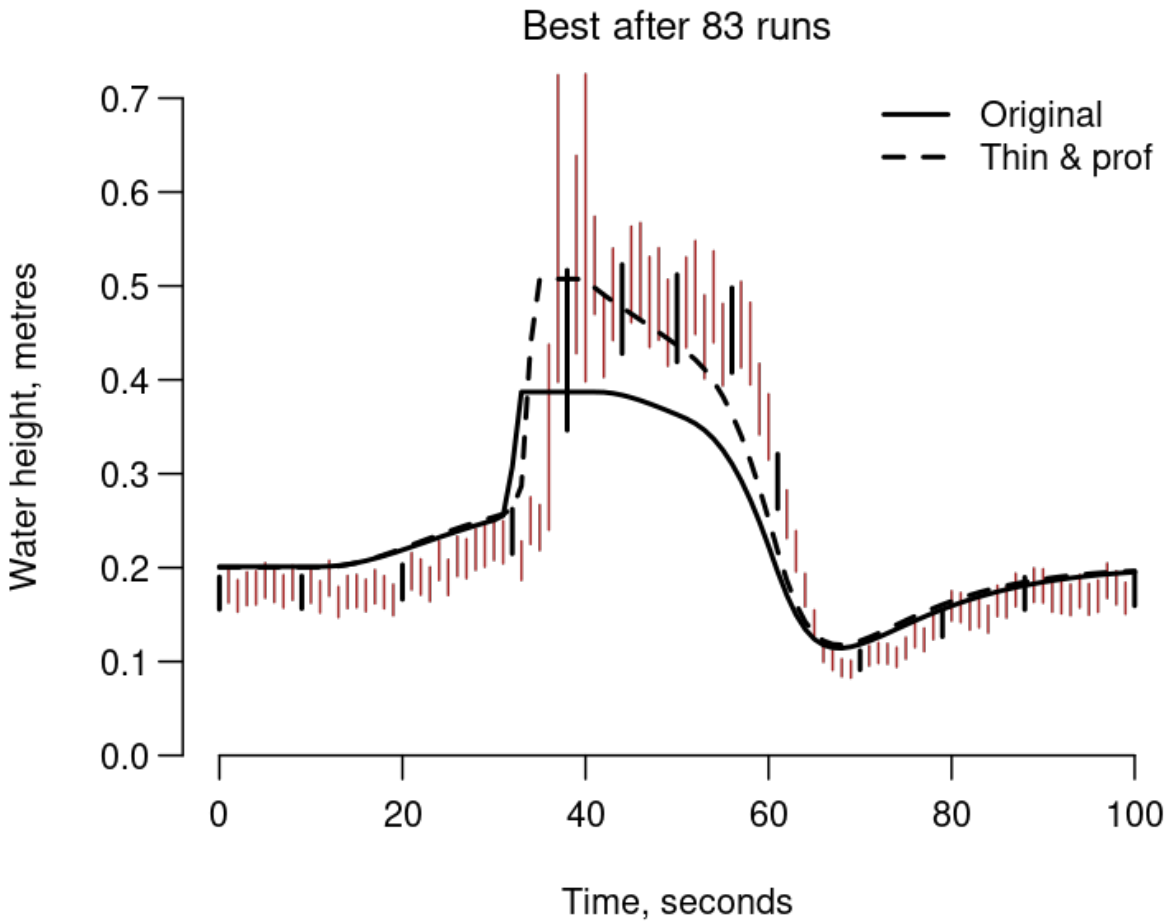
Second phase of runs

Original continues with Downhill Simplex. Thin & prof switches to Bayesian optimization.



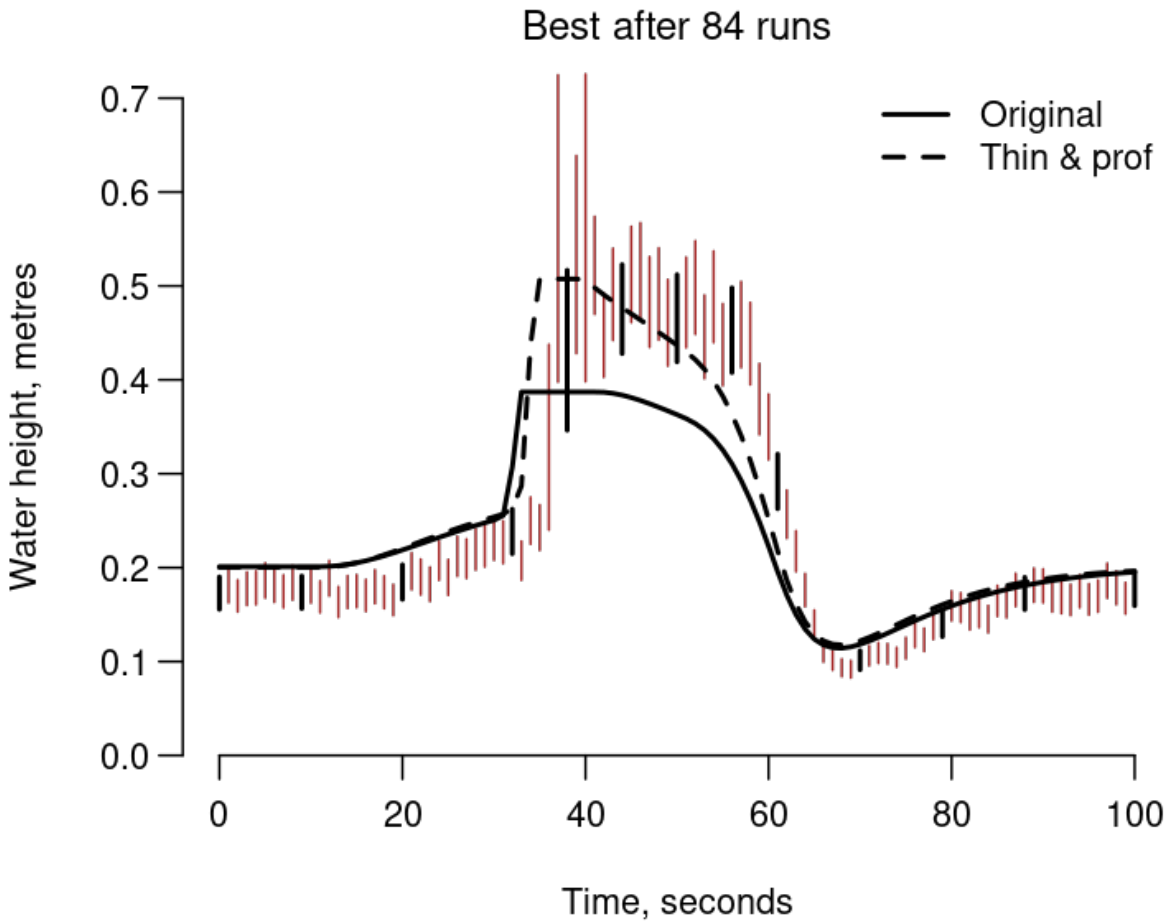
Second phase of runs

Original continues with Downhill Simplex. Thin & prof switches to Bayesian optimization.



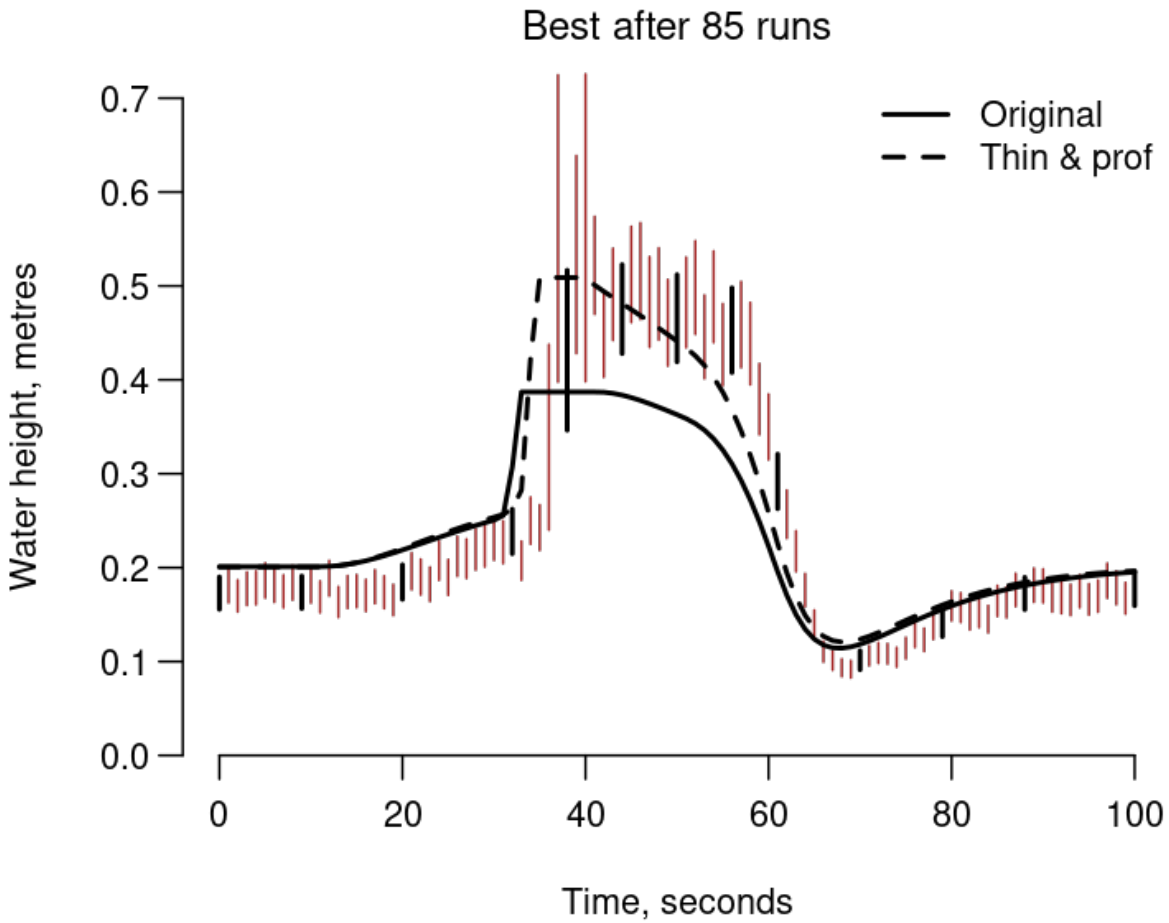
Second phase of runs

Original continues with Downhill Simplex. Thin & prof switches to Bayesian optimization.



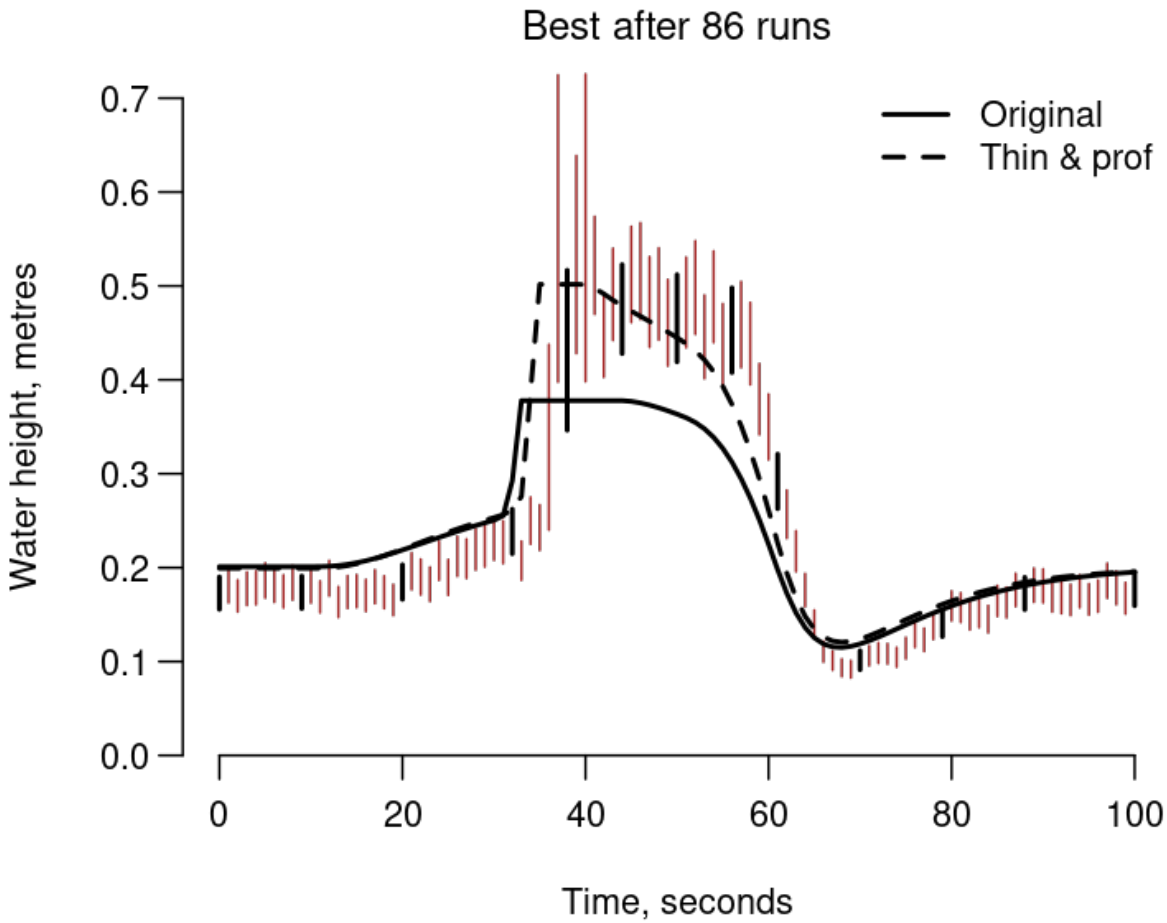
Second phase of runs

Original continues with Downhill Simplex. Thin & prof switches to Bayesian optimization.



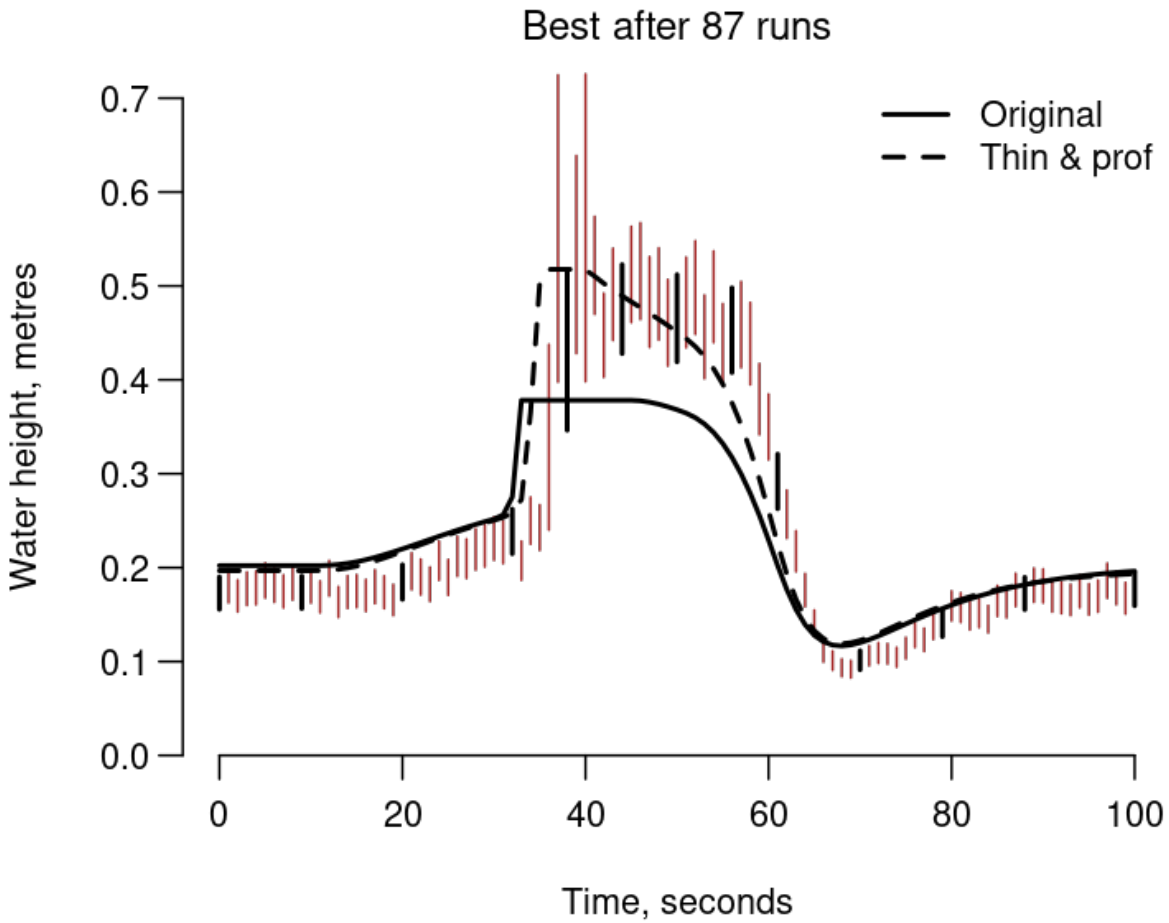
Second phase of runs

Original continues with Downhill Simplex. Thin & prof switches to Bayesian optimization.



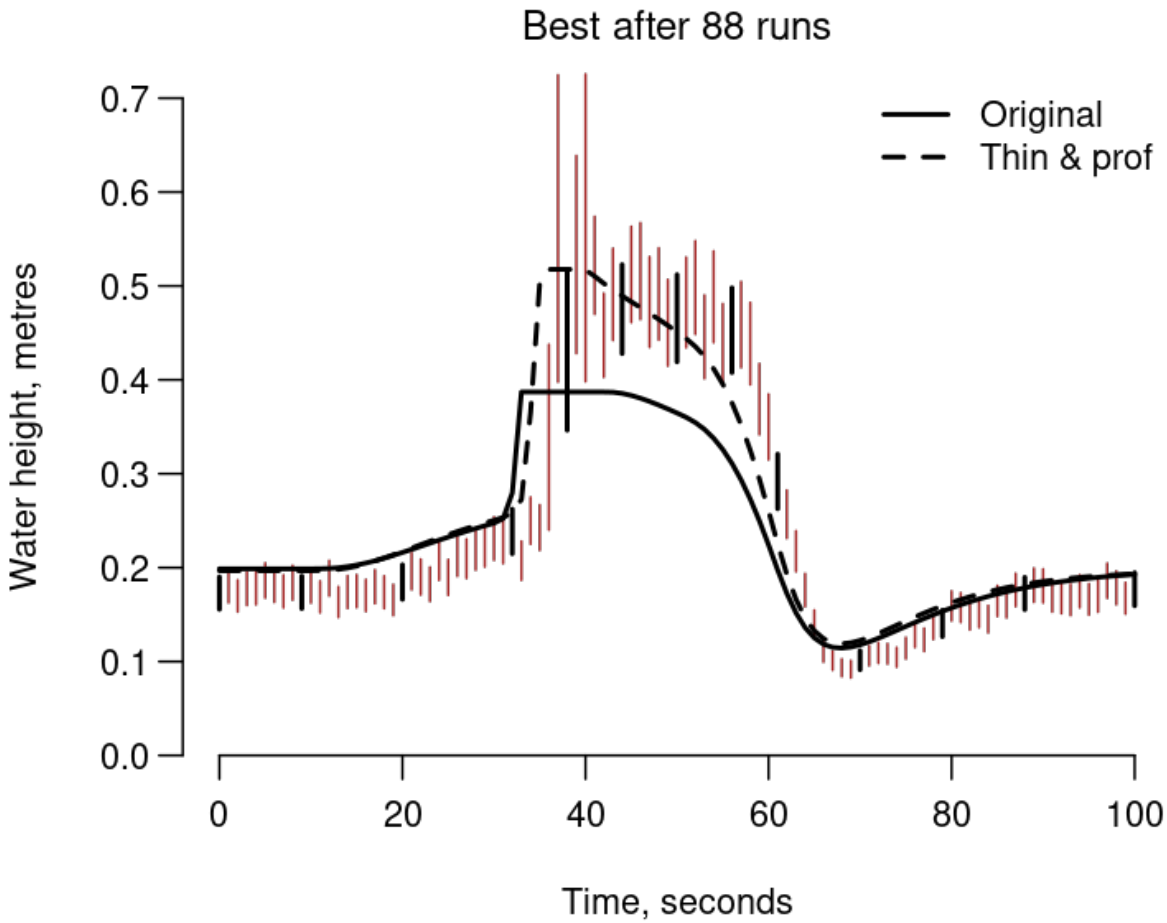
Second phase of runs

Original continues with Downhill Simplex. Thin & prof switches to Bayesian optimization.



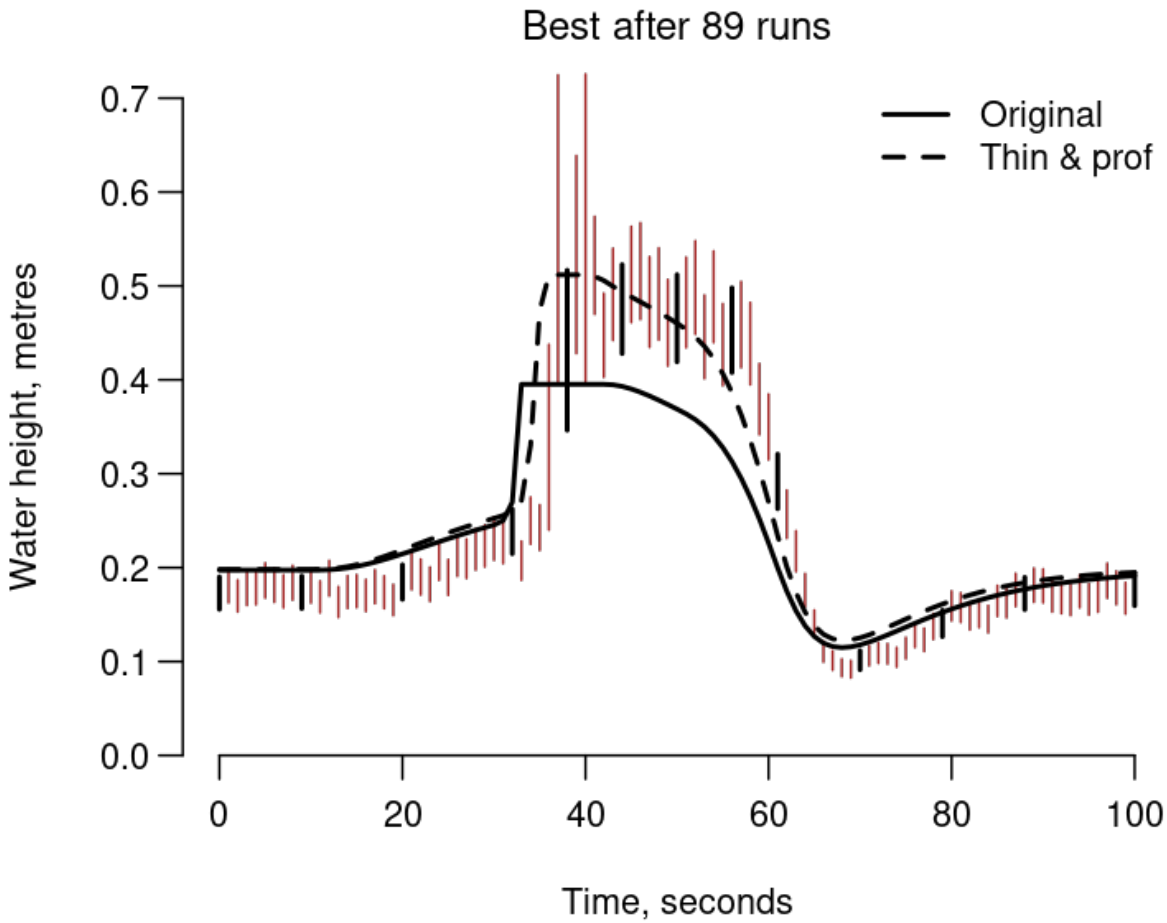
Second phase of runs

Original continues with Downhill Simplex. Thin & prof switches to Bayesian optimization.



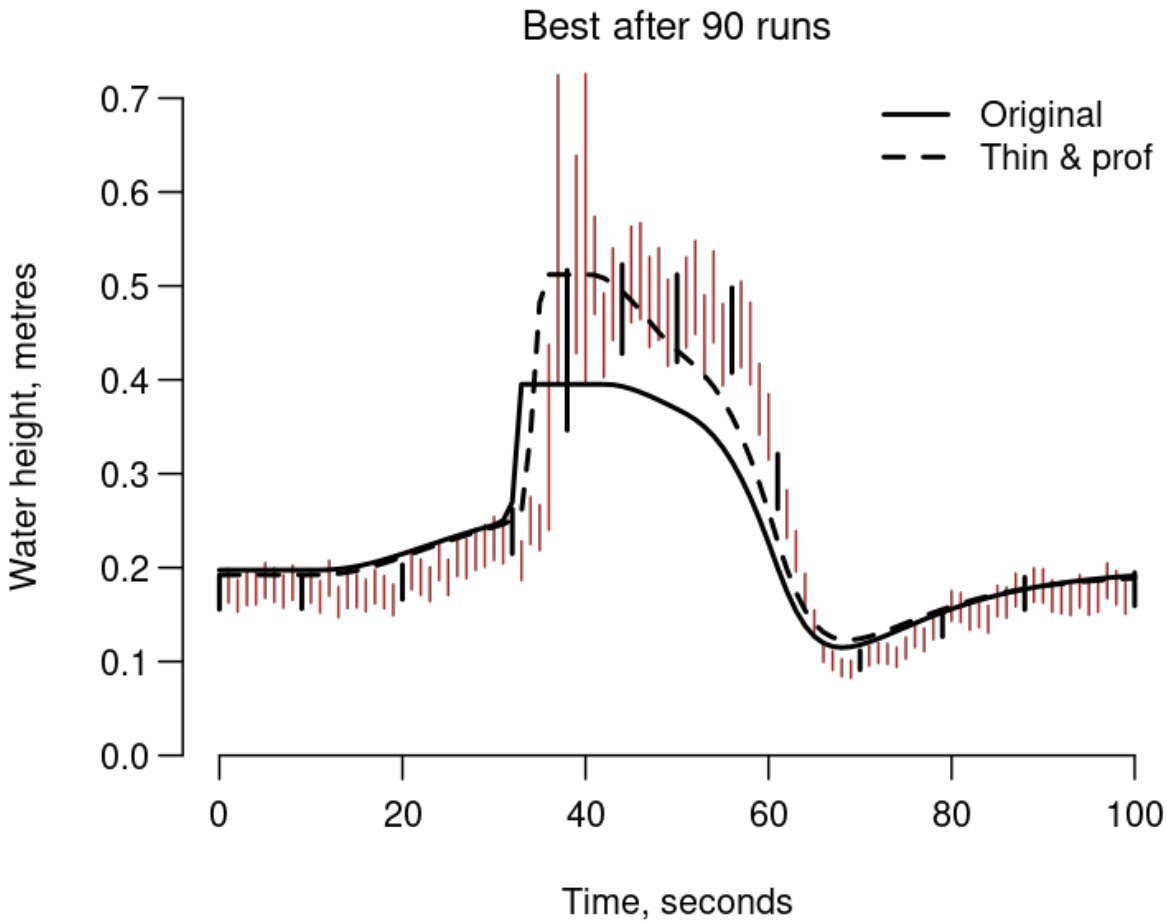
Second phase of runs

Original continues with Downhill Simplex. Thin & prof switches to Bayesian optimization.



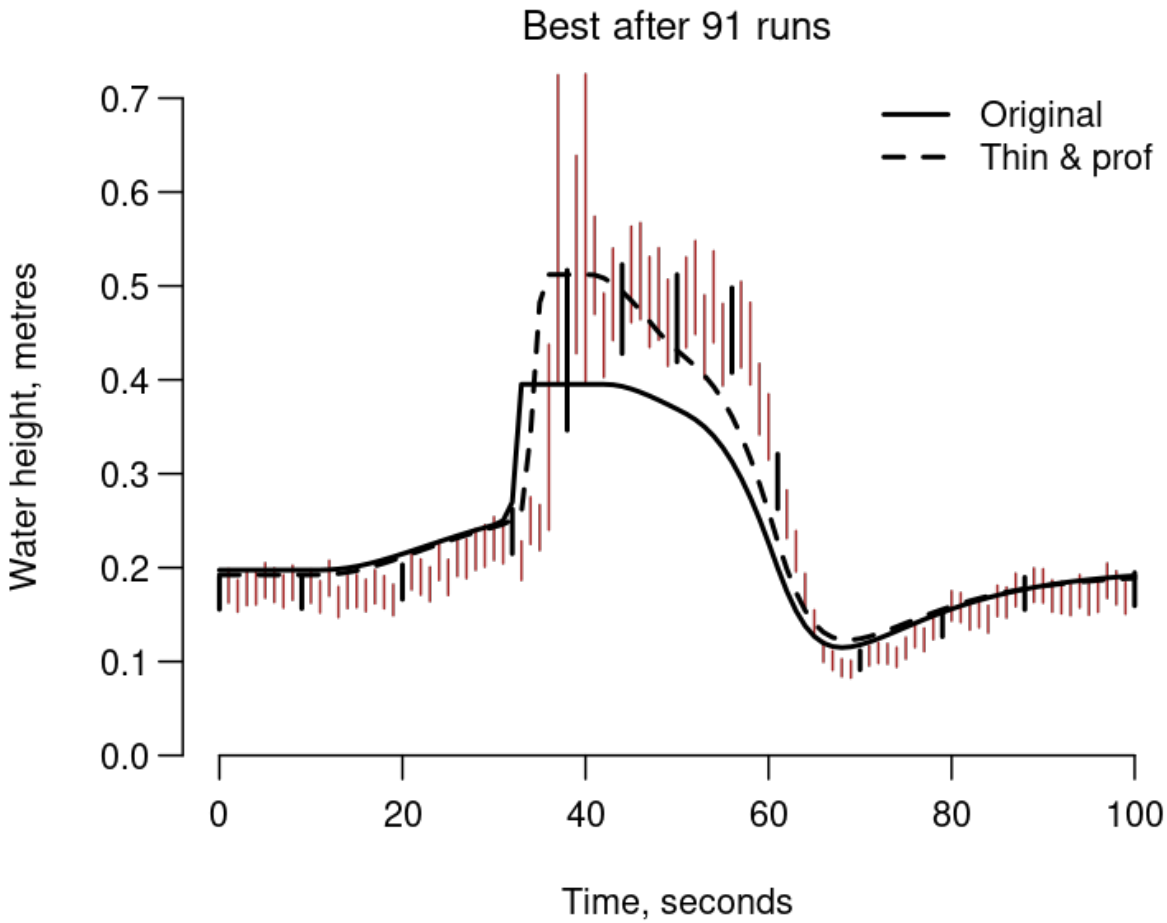
Second phase of runs

Original continues with Downhill Simplex. Thin & prof switches to Bayesian optimization.



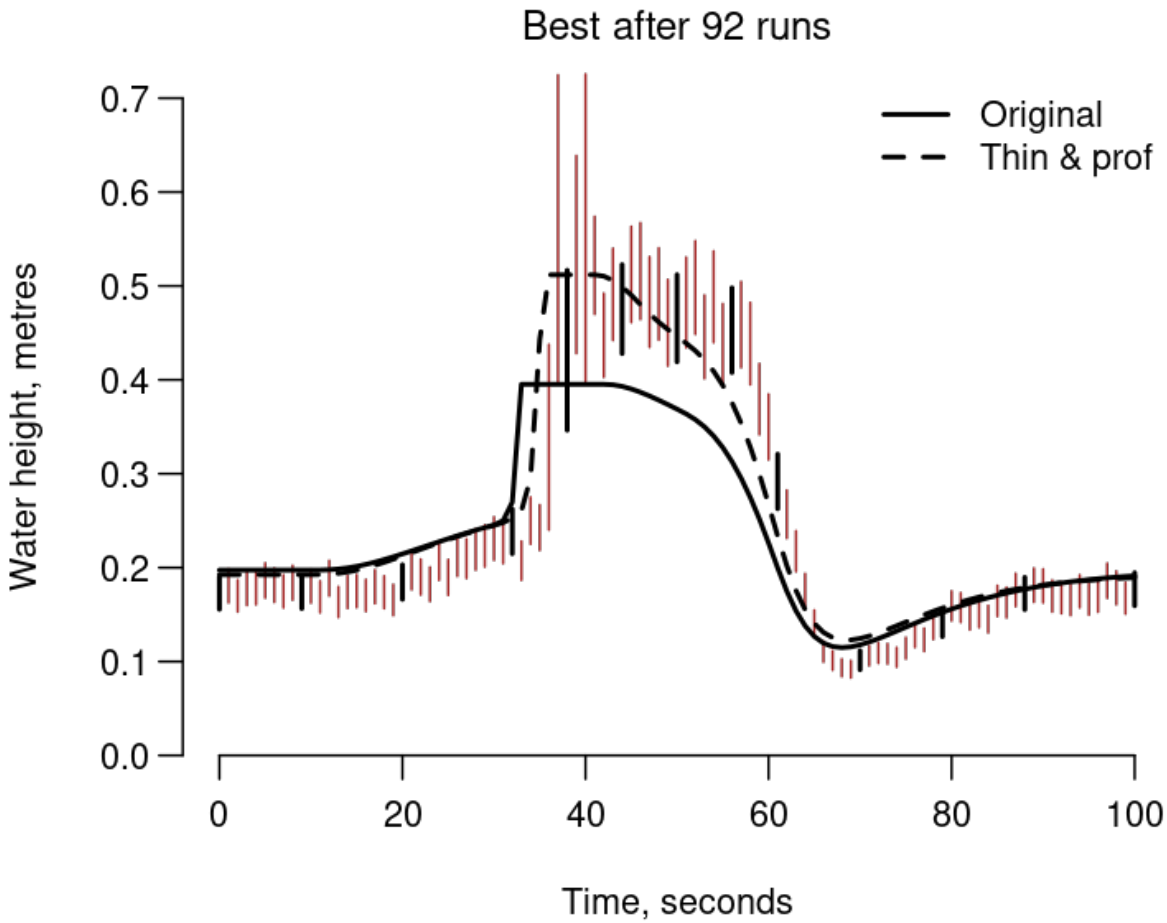
Second phase of runs

Original continues with Downhill Simplex. Thin & prof switches to Bayesian optimization.



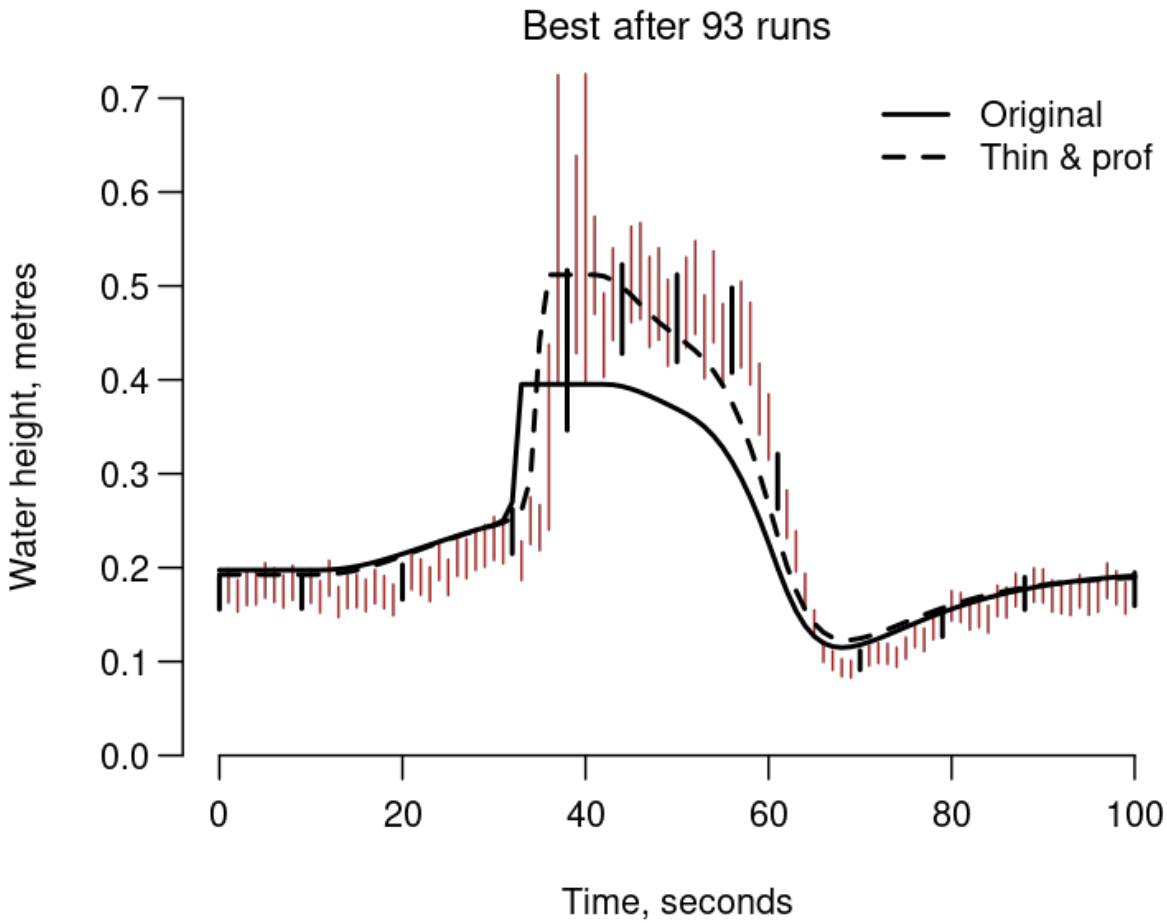
Second phase of runs

Original continues with Downhill Simplex. Thin & prof switches to Bayesian optimization.



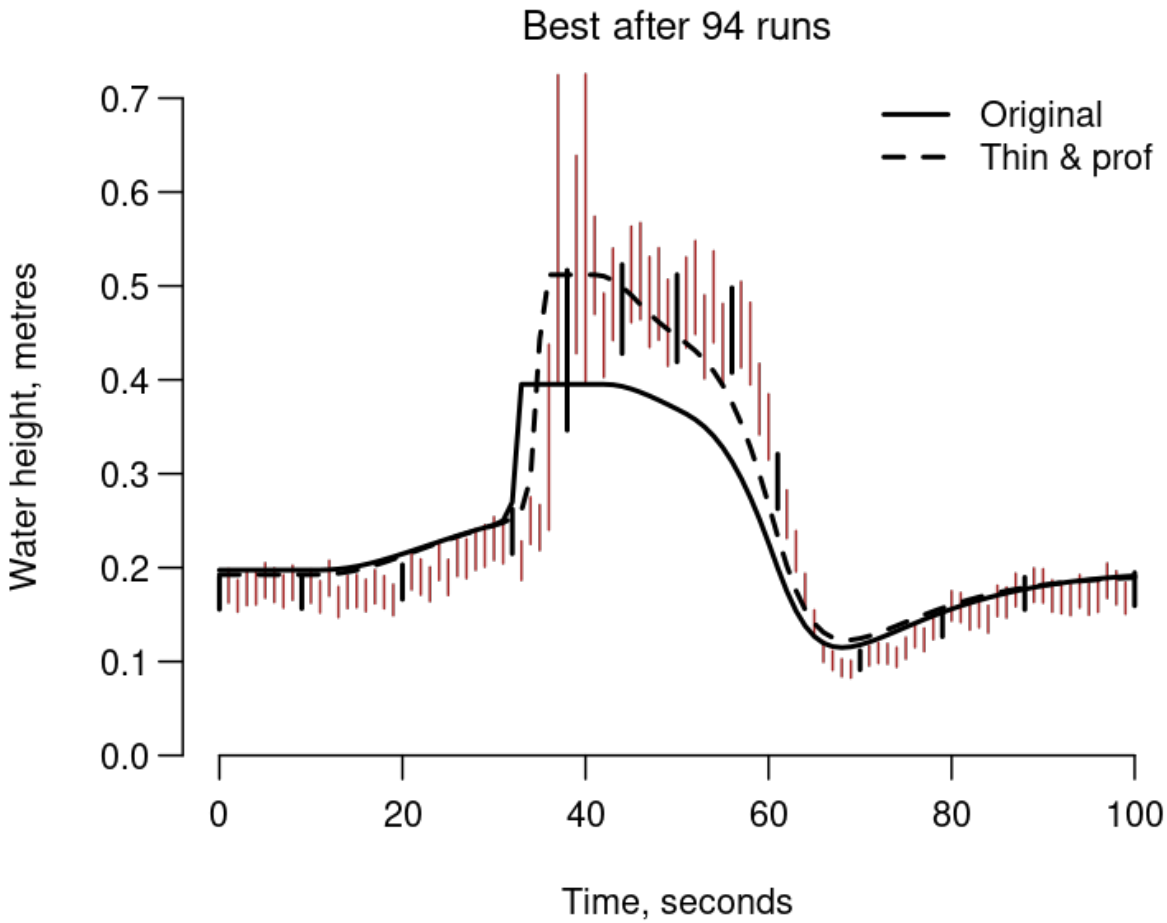
Second phase of runs

Original continues with Downhill Simplex. Thin & prof switches to Bayesian optimization.



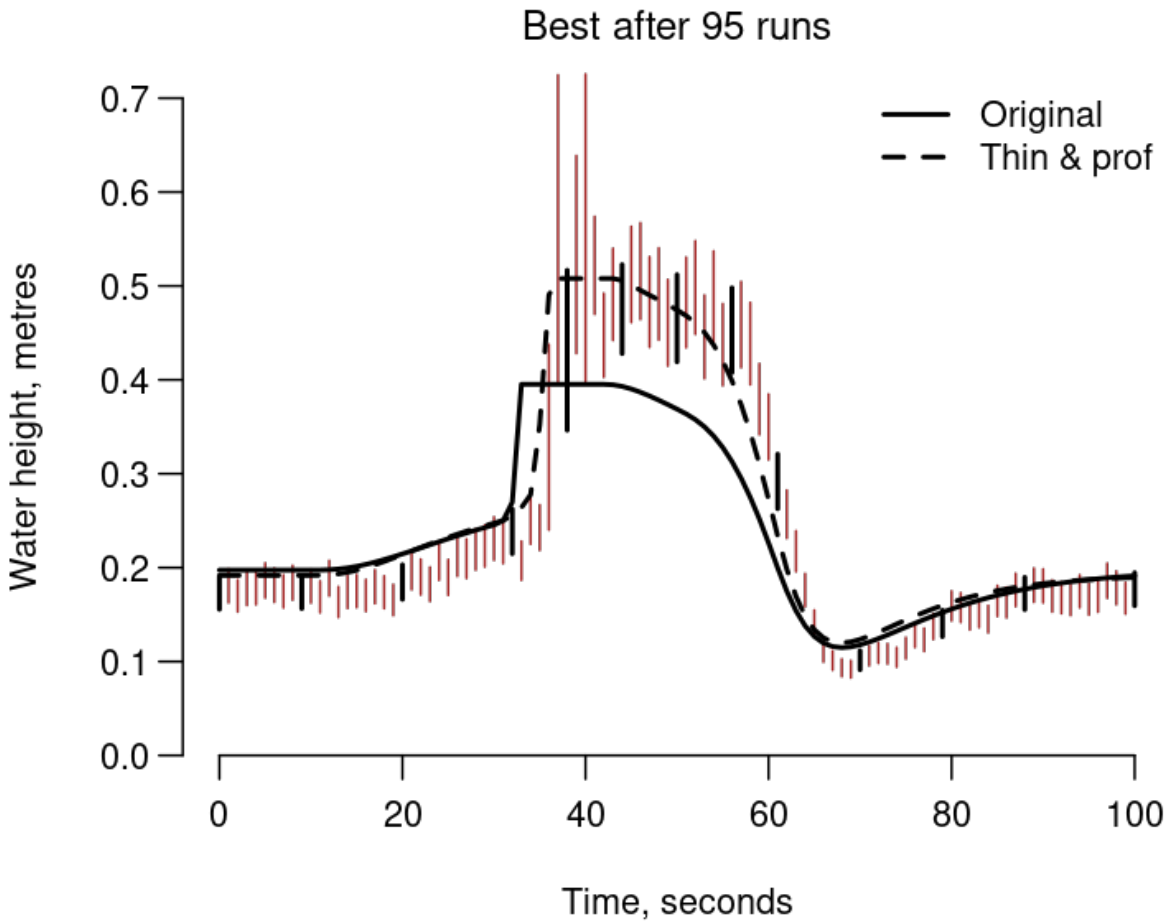
Second phase of runs

Original continues with Downhill Simplex. Thin & prof switches to Bayesian optimization.



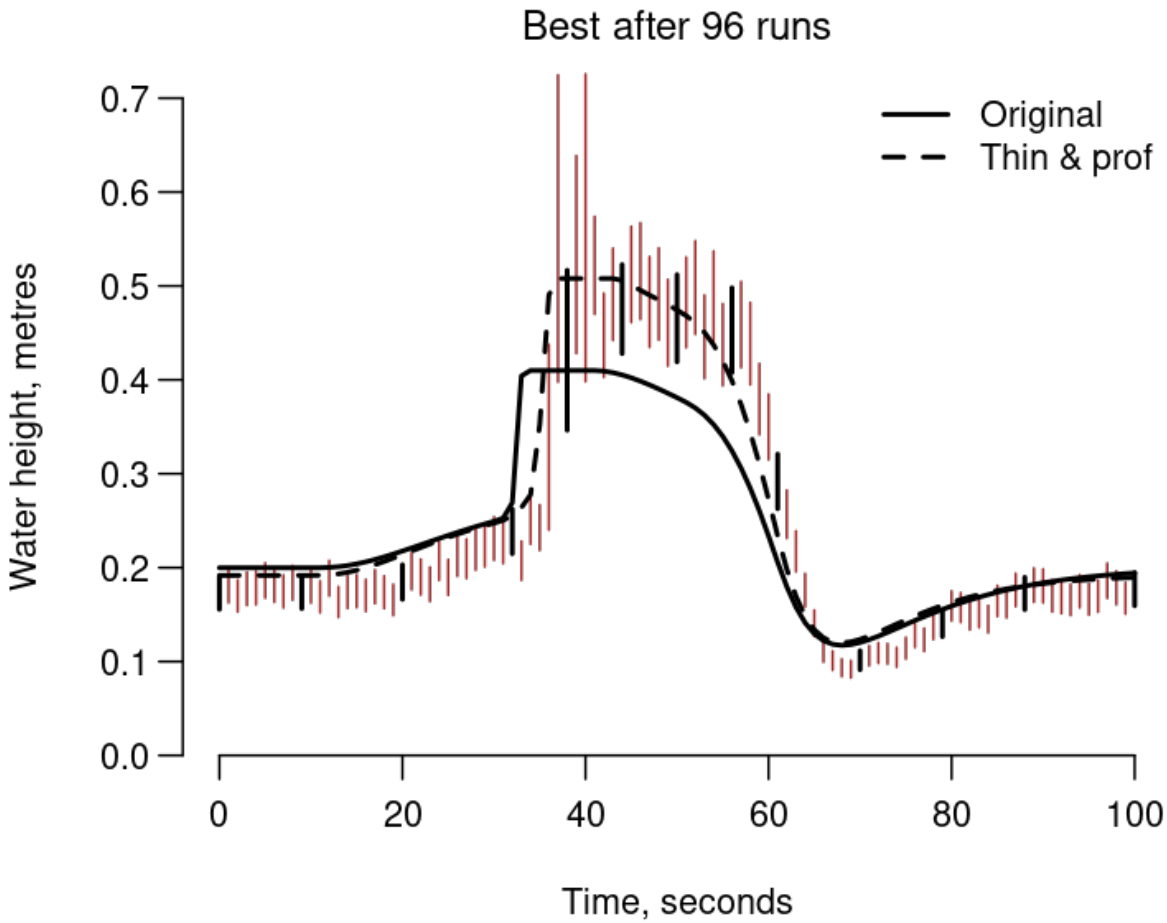
Second phase of runs

Original continues with Downhill Simplex. Thin & prof switches to Bayesian optimization.



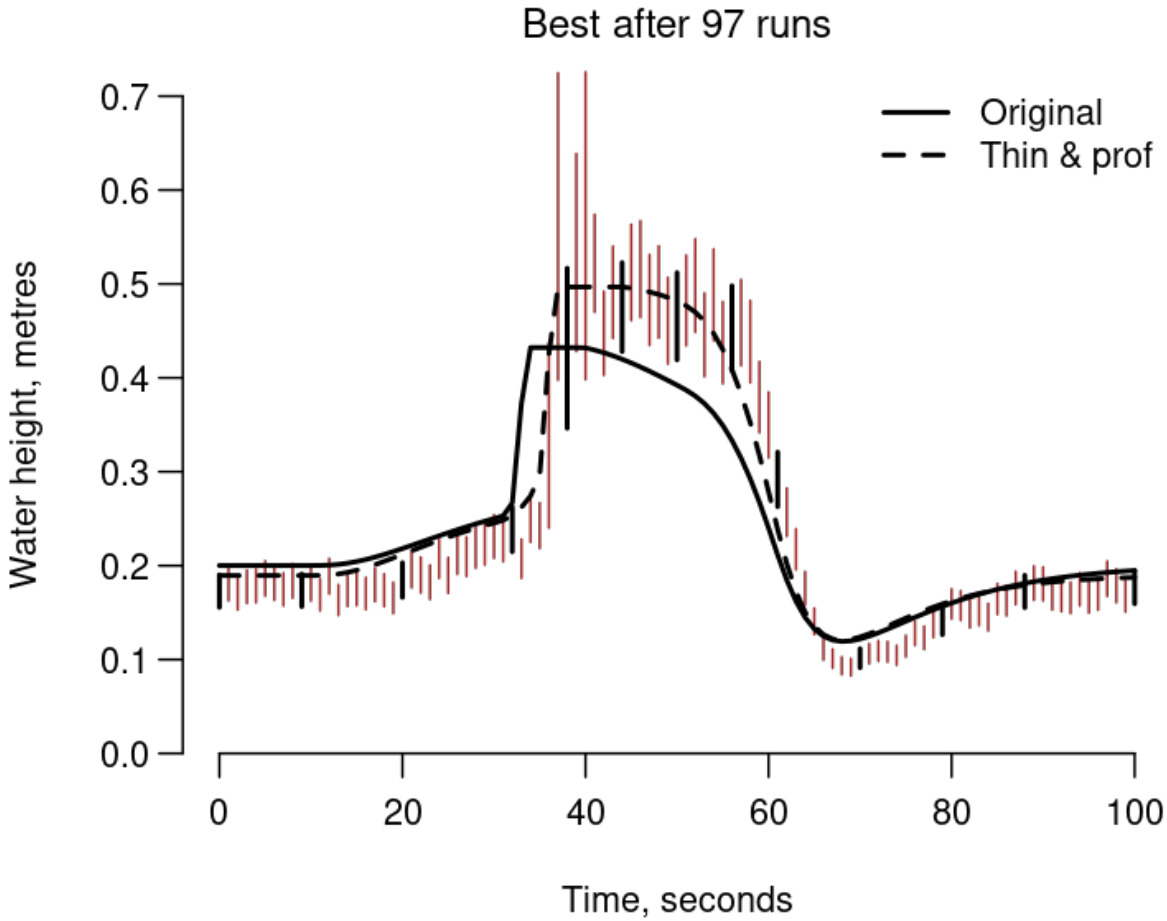
Second phase of runs

Original continues with Downhill Simplex. Thin & prof switches to Bayesian optimization.



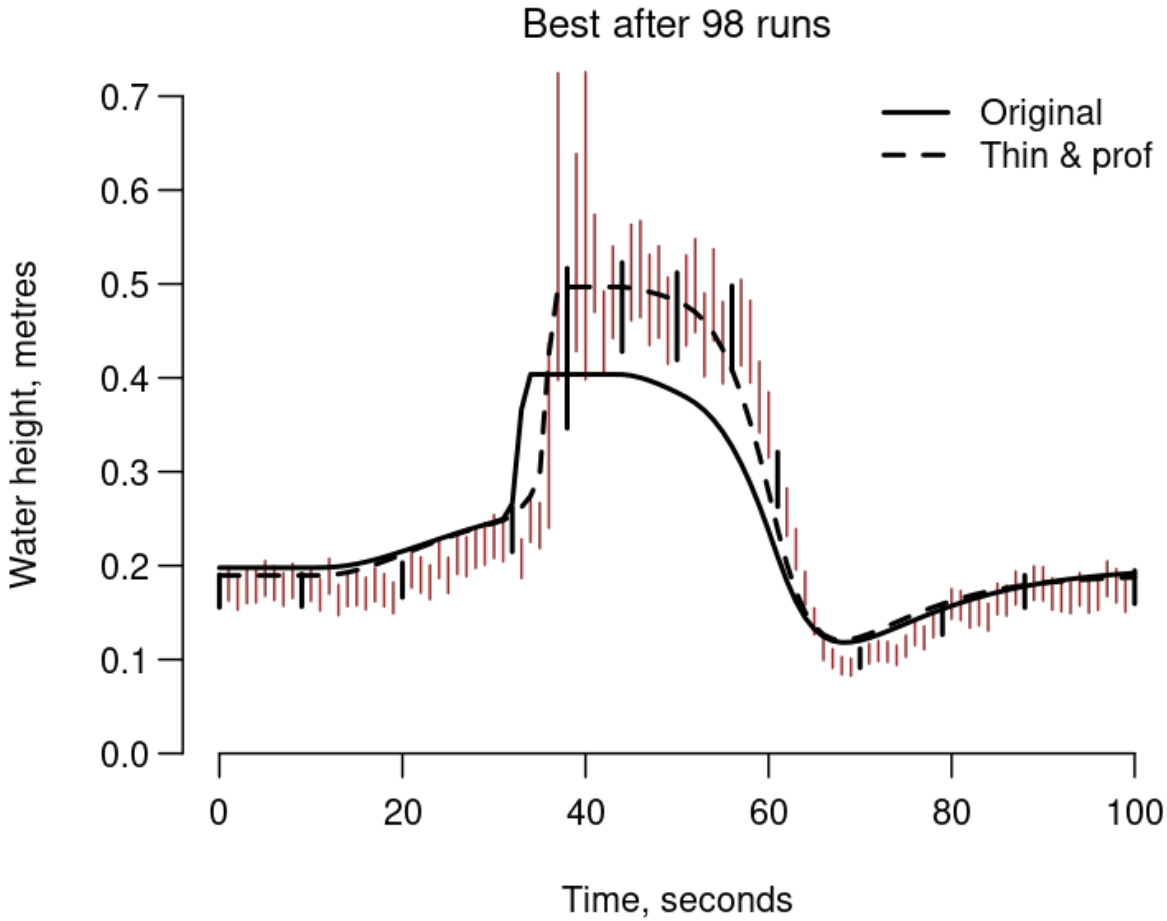
Second phase of runs

Original continues with Downhill Simplex. Thin & prof switches to Bayesian optimization.



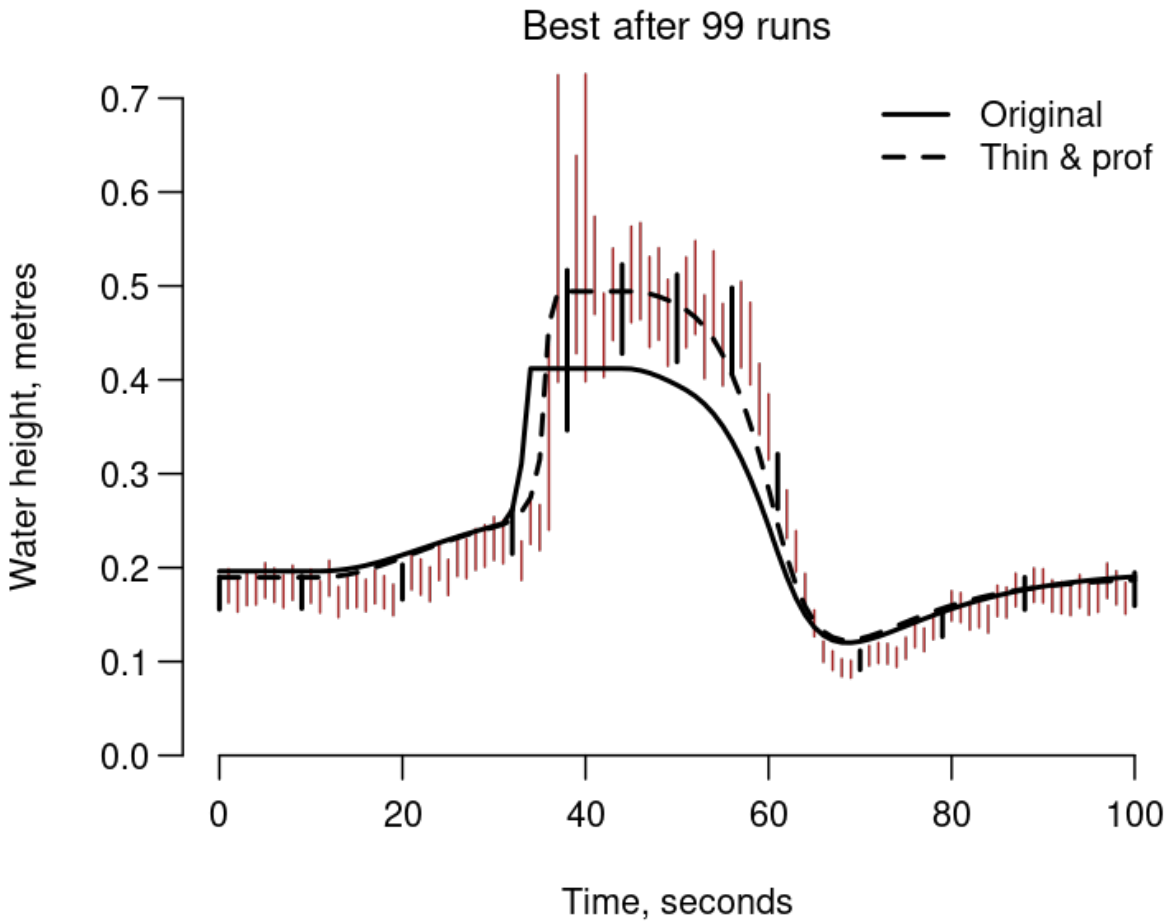
Second phase of runs

Original continues with Downhill Simplex. Thin & prof switches to Bayesian optimization.



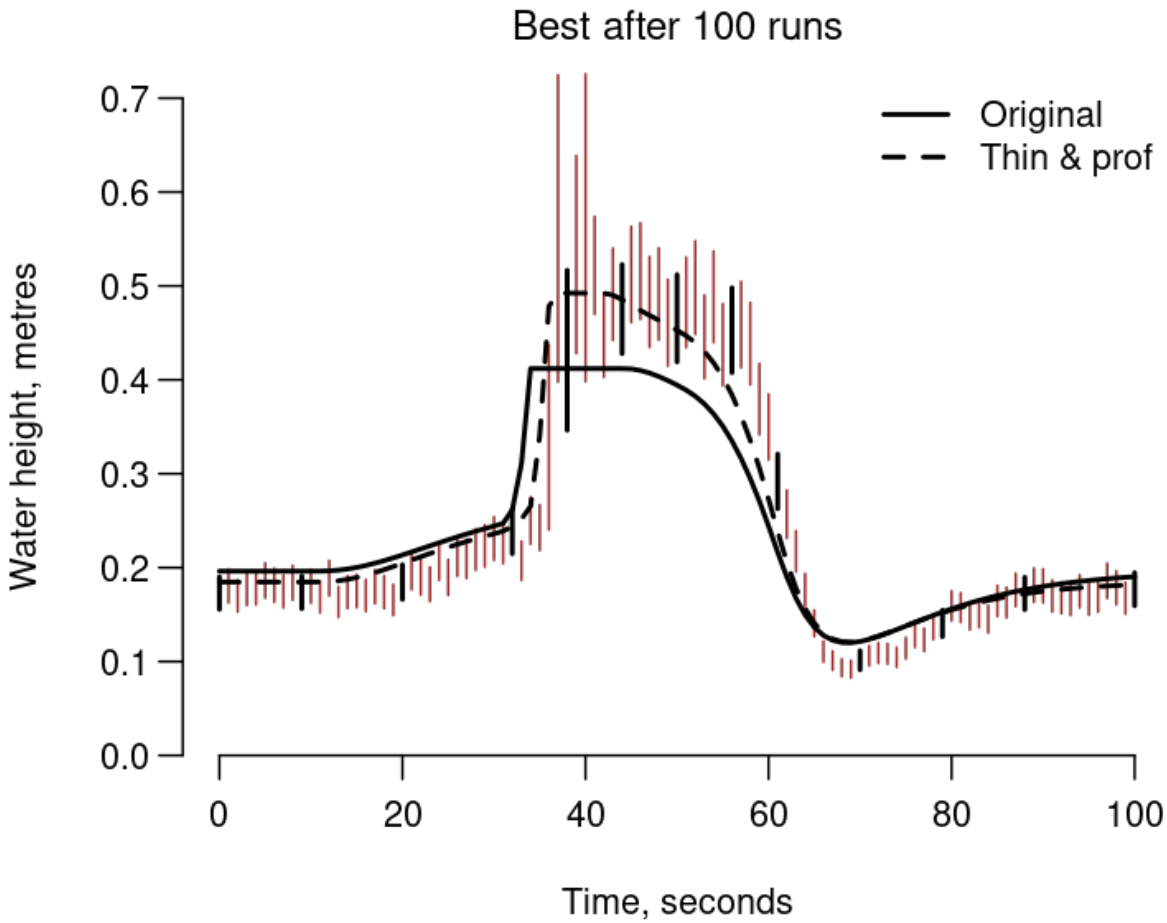
Second phase of runs

Original continues with Downhill Simplex. Thin & prof switches to Bayesian optimization.



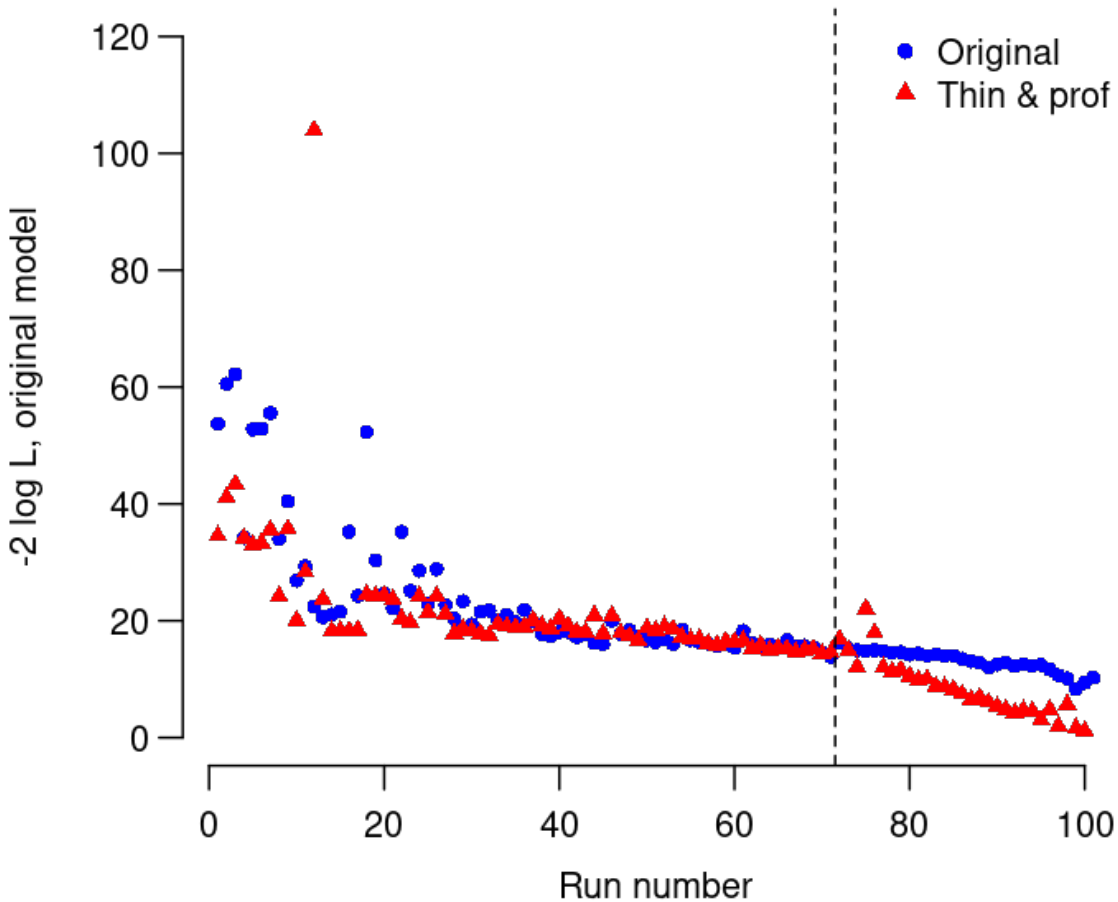
Second phase of runs

Original continues with Downhill Simplex. Thin & prof switches to Bayesian optimization.



A gratifying result

The optimized thin & prof model typically performs better than the optimized original model, *even according to the original model log-likelihood* (α plugged in).



Extensions

Other structural outputs; e.g., spatial

Exactly the same approach: thin and profile.

Two or more time-series

Need to choose one of:

1. Same α and κ for both/all time-series?
2. Different α, same κ?
3. Same α, different κ? (this one seems odd)
4. Different α, different κ?

Prediction

As well as θ^*, need to carry information about α and κ through into the prediction: possibly just plug-in. We get pointwise approximate 95% confidence intervals for the predicted time-series under new forcing.

Summary

1. We recognize the limitations of the simulator by including an offset and a discrepancy. Not recognizing this results in a log-likelihood function which is lumpy.

Summary

1. We recognize the limitations of the simulator by including an offset and a discrepancy. Not recognizing this results in a log-likelihood function which is lumpy.

2. By thinning the observations we can profile out the offset and the discrepancy variance, and the resulting likelihood function is smooth-ish.

Summary

1. We recognize the limitations of the simulator by including an offset and a discrepancy. Not recognizing this results in a log-likelihood function which is lumpy.
2. By thinning the observations we can profile out the offset and the discrepancy variance, and the resulting likelihood function is smooth-ish.
3. Downhill Simplex optimization then works fairly well, and generates a set of candidate points that – we hope – straddle the global optimum.

Summary

1. We recognize the limitations of the simulator by including an offset and a discrepancy. Not recognizing this results in a log-likelihood function which is lumpy.
2. By thinning the observations we can profile out the offset and the discrepancy variance, and the resulting likelihood function is smooth-ish.
3. Downhill Simplex optimization then works fairly well, and generates a set of candidate points that – we hope – straddle the global optimum.
4. We switch to Bayesian optimization to finish the job efficiently, emulating the log-likelihood with a Gaussian Process, and using adaptive search.

Summary

1. We recognize the limitations of the simulator by including an offset and a discrepancy. Not recognizing this results in a log-likelihood function which is lumpy.
2. By thinning the observations we can profile out the offset and the discrepancy variance, and the resulting likelihood function is smooth-ish.
3. Downhill Simplex optimization then works fairly well, and generates a set of candidate points that – we hope – straddle the global optimum.
4. We switch to Bayesian optimization to finish the job efficiently, emulating the log-likelihood with a Gaussian Process, and using adaptive search.
5. While there are no guarantees, we hope this approach produces a better estimate of the best input, for the same number of simulator runs.

T H E E N D

Time for questions and discussion.

References

Goldstein, M. and Rougier, J. C. (2004). Probabilistic formulations for transferring inferences from mathematical models to physical systems. *SIAM Journal on Scientific Computing*, 26(2):467–487.

Goldstein, M. and Rougier, J. C. (2009). Reified Bayesian modelling and inference for physical systems. *Journal of Statistical Planning and Inference*, 139:1221–1239. With discussion, pp. 1243–1256.

Gu, M., Wang, X., and Berger, J. O. (2018). Robust Gaussian stochastic process emulation. *Annals of Statistics*, 46(6A):3038–3066.

Osborne, M. A., Garnett, R., and Roberts, S. J. (2009). Gaussian processes for global optimization. *3rd International Conference on Learning and Intelligent Optimization (LION3)*, pages 1–15. Available at `http://www.robots.ox.ac.uk/~mosb/public/pdf/1419/Osborne%20et%20al.%20-%202009%20-%20Gaussian%20processes%20for%20global%20optimization.pdf`.

Rougier, J. C. (2007). Probabilistic inference for future climate using an ensemble of climate model evaluations. *Climatic Change*, 81:247–264.

Rougier, J. C. (2013). 'Intractable and unsolved': Some thoughts on statistical data assimilation with uncertain static parameters. *Phil. Trans. R. Soc. A*, 371:20120297.

Rougier, J. C. and Goldstein, M. (2014). Climate simulators and climate projections. *Annual Review of Statistics and Its Application*, 1:103–123.

The 'best input' model

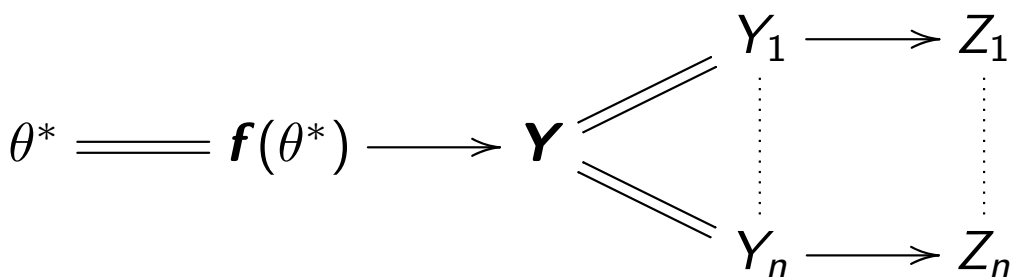
See Goldstein and Rougier (2004, 2009), Rougier (2007), Rougier and Goldstein (2014). This is the ubiquitous model for linking simulator runs, the system, and observables.

- There is a 'best' value of the parameters, θ^*, such that $\boldsymbol{Y} \perp\!\!\!\perp \theta^* \mid \boldsymbol{f}(\theta^*)$, or, as a DAG,

$$\theta^* \Longrightarrow \boldsymbol{f}(\theta^*) \longrightarrow \boldsymbol{Y}$$

  where $\Longrightarrow$ denotes a deterministic edge.
  If we knew θ^*, we'd run the simulator just once to predict $\boldsymbol{Y}$, no matter what the value of θ^* happened to be.

- Also ubiquitous is to add on a simple error structure for the observables



Return

Can we do better? (cont)

- We can profile out α directly

$$L_{\mathcal{J}}(\theta, \kappa) := \max_{\alpha \in [-\alpha_{\max}, \alpha_{\max}]} L_{\mathcal{J}}(\theta, \alpha, \kappa) = L_{\mathcal{J}}(\theta, \tilde{\alpha}(\theta, \kappa), \kappa)$$

where

$$\hat{\alpha}(\theta, \kappa) := \sum_{i \in \mathcal{J}} w_i (z_i^{\text{obs}} - f_i(\theta)), \quad w_i := \frac{(\kappa^2 + \sigma_i^2)^{-1}}{\sum_j (\kappa^2 + \sigma_j^2)^{-1}}$$

$$\tilde{\alpha}(\theta, \kappa) := -\alpha_{\max} \vee \hat{\alpha}(\theta, \kappa) \wedge \alpha_{\max}.$$

- Then we can profile out κ using a 1D numerical optimization,

$$L_{\mathcal{J}}(\theta) := \max_{\kappa \in [0, \kappa_{\max}]} L_{\mathcal{J}}(\theta, \tilde{\alpha}(\theta, \kappa), \kappa).$$

- Computing $L_{\mathcal{J}}(\theta)$ only requires one run of the simulator, plus a quick numerical optimization; i.e., its cost is comparable to computing the original $L(\theta)$.

Can we do better? (cont)

- The bounds on α and κ in the profile likelihoods are not just for show. Profile likelihood is a notoriously tricky approach for estimating variances, and we want to keep α and κ fairly close to their default values of 0 and 0 in order to stop the profile running off to a statistical but not plausible solution.

  I have been using

$$\alpha_{\max} = \kappa_{\max} = 2\,\text{median}\{\sigma_1, \ldots, \sigma_n\}.$$

- In summary, we have (at least) two approaches:

  1. Original model, which is just the scaled sum of squared deviations over all outputs, i.e. no discrepancy.
  2. Thin & prof model, where the outputs have been thinned, and the discrepancy parameters have been profiled out.

Return

Bayesian optimization (cont)

A myopic adaptive approach (simple, room for improvement):

0a. Find the bounding box of all inputs so far, $\mathcal{B}$.

0b. Expand by 10% from the centroid to give $\mathcal{B}^+$ (don't overshoot the parameter limits).

0c. Fill $\mathcal{B}^+$ with a grid to give $\mathcal{S}^+$.

1. Build a GP emulator of ℓ using all runs so far.
2. Evaluate $\mathbb{E}\{\lambda(L_\theta)\}$ at every point in $\mathcal{S}^+$.
3. Improve the best point on $\mathcal{S}^+$ using a quasi-Newton method, staying inside $\mathcal{B}^+$, to give θ^{new}.
4. Run the simulator at θ^{new} and compute the profile likelihood $\ell_{\theta^{\text{new}}}$.
5. If θ^{new} is outside $\mathcal{B}$, go back to 0, otherwise go back to 1.

▸ Return