VENTANA systems, inc.

Vensim® Software

Linking systems thinking to powerful dynamic models

Calibration with Vensim

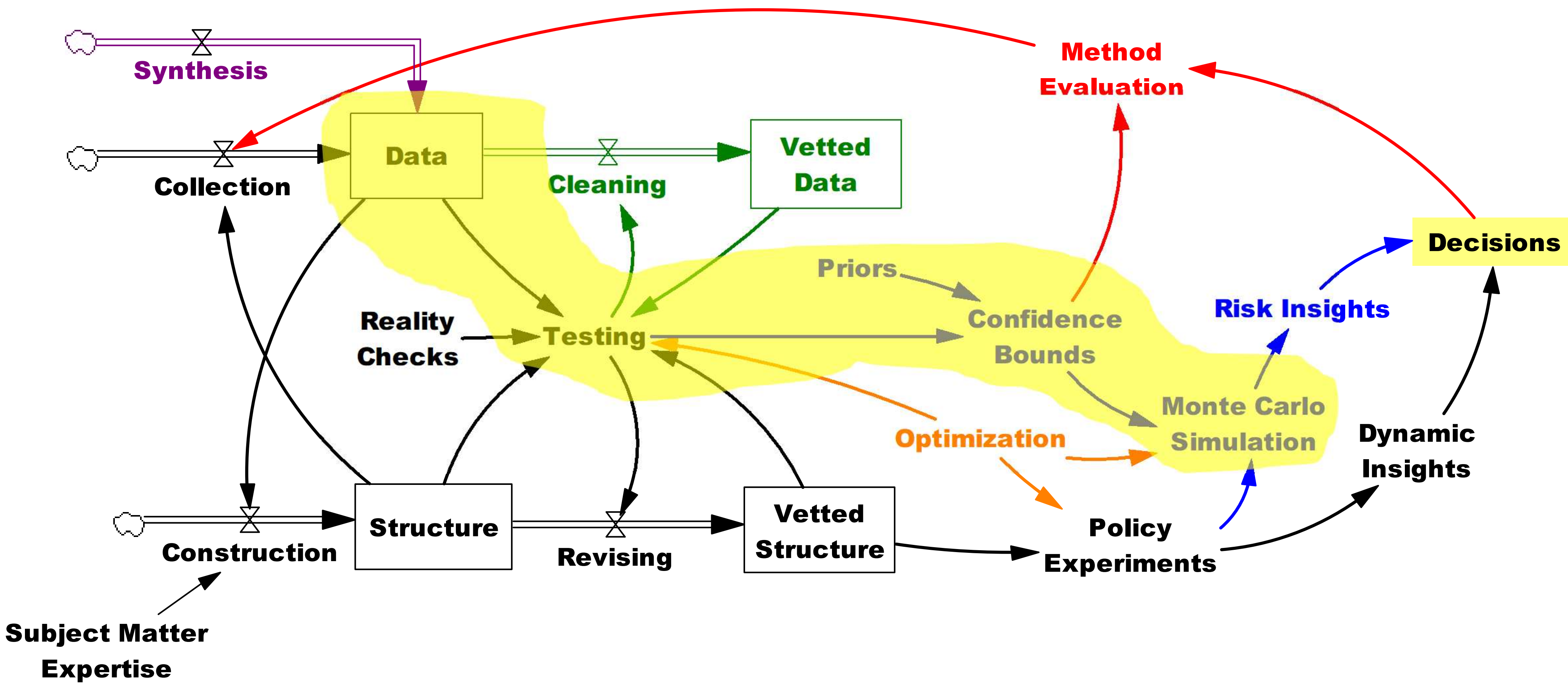
Tom Fiddaman

2022

Agenda

- **Introduction**
- **Synthetic data**
- **Naïve calibration**
- **Maximum likelihood**
- **Markov Chain Monte Carlo (MCMC)**
- **Kalman filtering (briefly)**

The Big Picture



Overview

- **Lotka-Volterra Predator-Prey model**

- **Generate synthetic data, by adding:**
  - Measurement error to the stocks of elk and wolves
  - Driving noise to the flows of births and mortality

- **Estimate parameters of the model from the data, by various methods**
  - Optionally, use mismatched structure (2nd order data-generating model, first order estimated model)

Caveats

In order to get done, we're approaching this problem a bit fast and loose. Be aware:

- **There is structural uncertainty as well as parameter uncertainty**
- **Statistics deserve deeper thought**
  - Weights
  - Covariance
  - Autocorrelation
  - Distributional assumptions
  - Measurement error & driving noise (Kalman filter)
- **We should be testing for multiple optima with multistart calibration runs**
- **Sample sizes for sensitivity and MCMC may be too small**

Other Pitfalls

- State dependent noise
- Sample size
- Data quality
- Autocorrelated errors
- Error covariance
- Measurement error
- State estimation
- Endogeneity

Calibration

- **Purposes**
  - Make better predictions or measurements
  - Reject models that can't replicate data (potentially a weak test of quality)
  - Learn about the model
  - Learn about the data
  - Provide face validity for reviewers
- **Closeness Measures**
  - Sum of squared errors & R^2
  - Mean Absolute Deviation
  - Mean Absolute Percent Error
  - Log Likelihood
- **Process**
  - Assume the model structure is right
    - If possible, test alternatives!
  - Simulate the model
    - Measure the closeness
    - Adjust the constants in the model
    - Iterate to improve
  - After convergence, evaluate the fit
    - Decide if the model needs revision
    - Investigate puzzles in the data

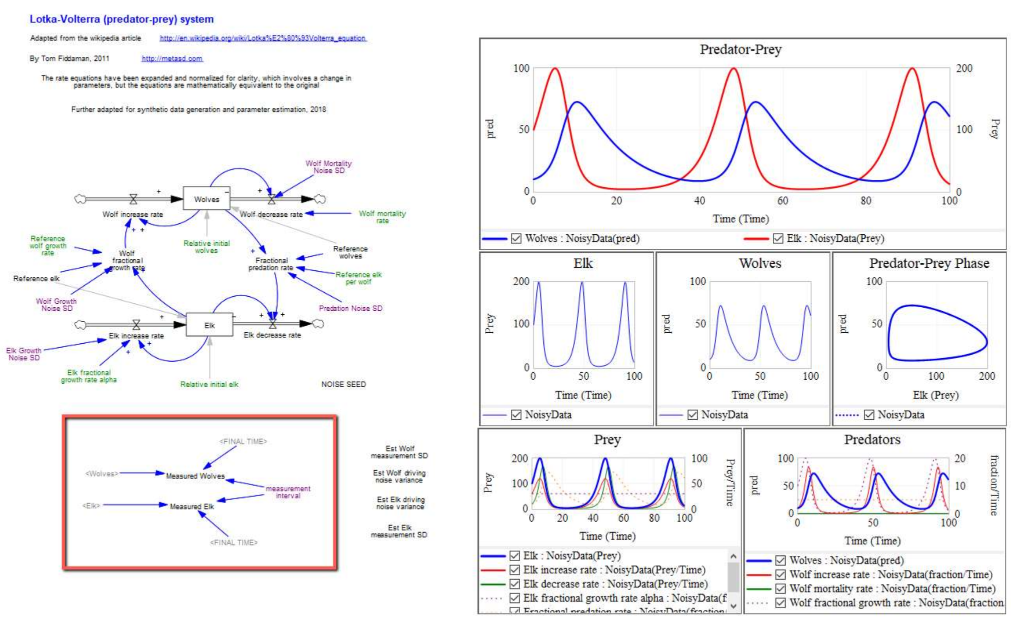
Model Tour

Synthetic Data Generation

- **Open the data-generating model**

Name	Status	Date modified	Type	Size
ElkWolves - estimate - kalman		7/23/2021 2:02 PM	File folder	
ElkWolves - estimate - mcmc		7/23/2021 2:06 PM	File folder	
ElkWolves - estimate - naive		7/23/2021 2:02 PM	File folder	
ElkWolves - estimate - start		7/23/2021 2:03 PM	File folder	
ElkWolves - estimate - weighted		7/23/2021 2:03 PM	File folder	
ElkWolves - generate		7/28/2021 7:17 PM	File folder	
ElkWolves - generate - 2o		7/23/2021 2:03 PM	File folder	

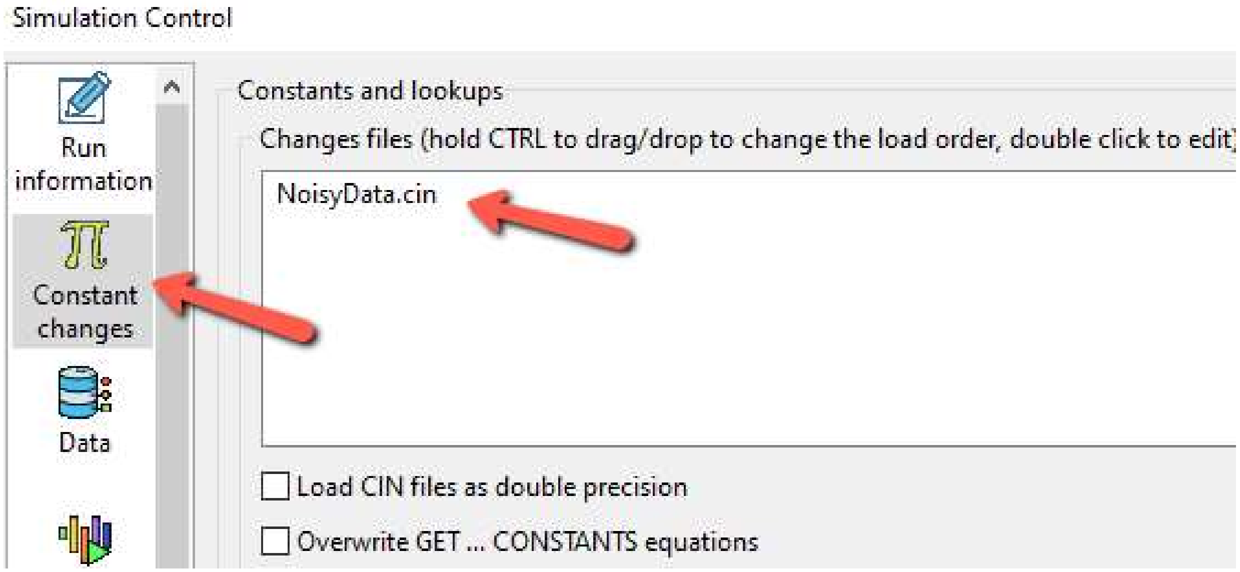
- **Run, and take a look at the "measured" variables (red box)**



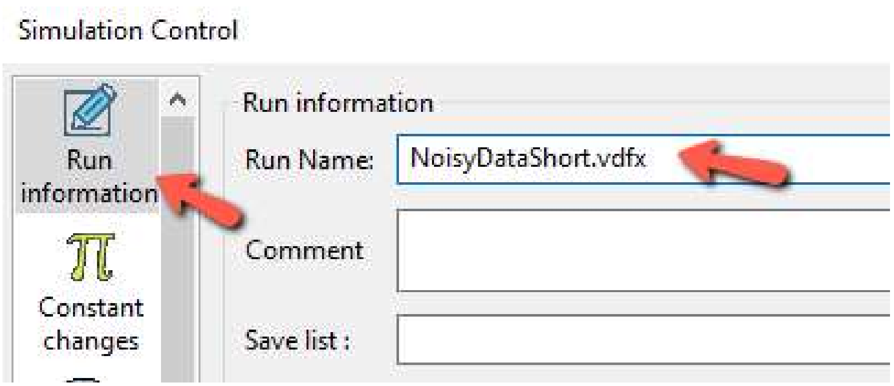
VENTANA **Vensim®**

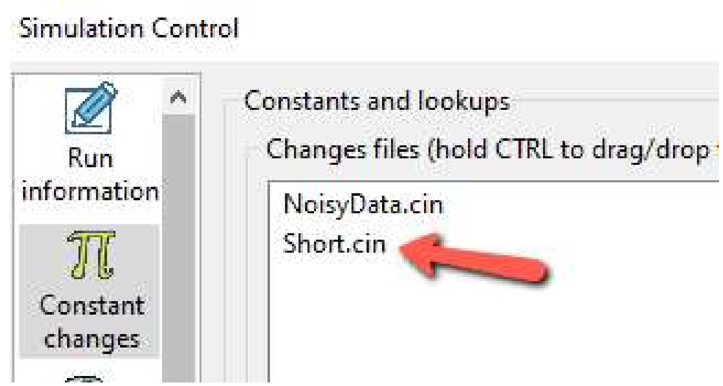
Synthetic Data Generation

- **Load "NoisyData.cin" and run the model**



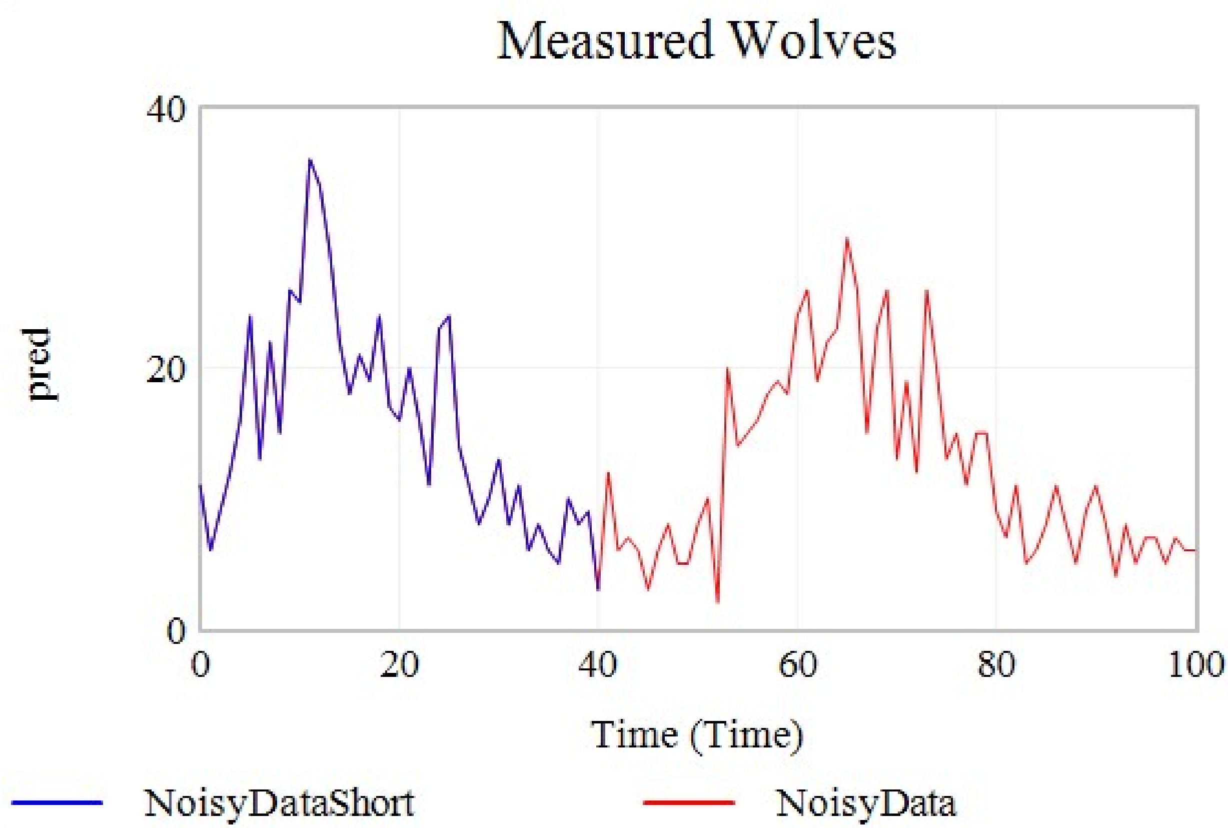
- **Run again, but add "short" to the run name, and load "Short.cin" to set FINAL TIME=40**





Synthetic Data Generation

- **This gives you 2 datasets**
  - Short time series, used to estimate parameters; export and re-import using a savelist to restrict the information to the measured state only
  - Long time series, for comparison of later estimates with “truth”



Naïve Calibration

- **Create a simple metric describing the distance of the model from the data**

- **Minimize the distance**

Mechanics – What We Need

- **Data**
  - A .vdf file, or
  - Equations with GET DIRECT DATA, GET XLS DATA
  - ODBC

- **A Payoff specifying what data series to match, and how to weight each one**

- **An Optimization Control file specifying**
  - which parameters to vary, and
  - what methods to use

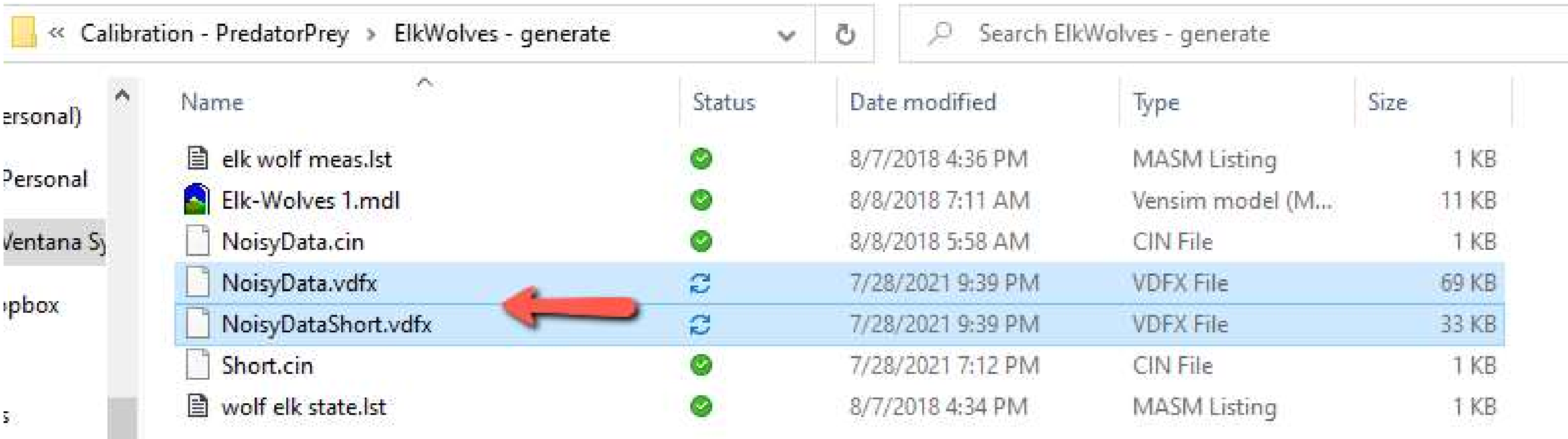
- **The optimizer then hill climbs to find the parameters that minimize the error between model and data**

Naïve Calibration
What do we get?

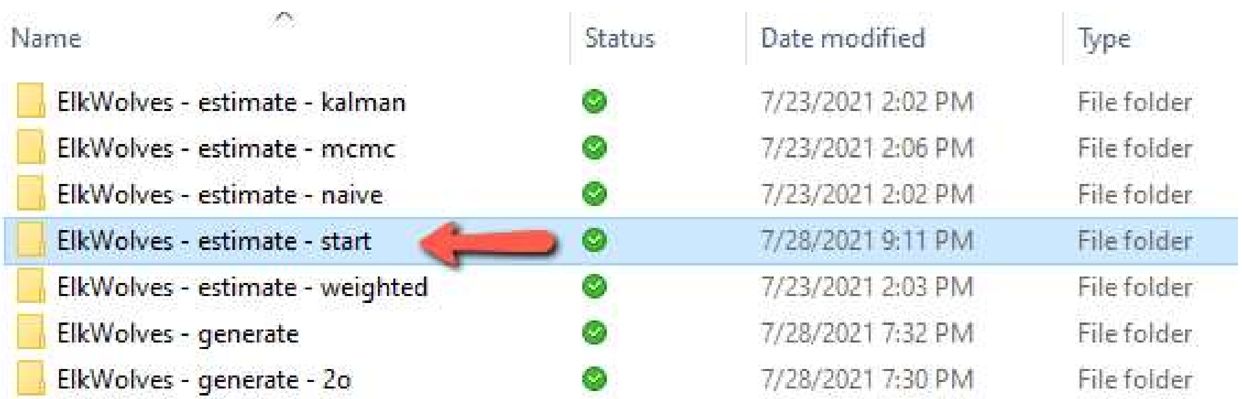
- **A run (.vdfx) with the best parameters**
- **An output file (.out) summarizing the parameters found**
  - Parameters can then be reused by loading the .out as a Changes file (like .cin files)
- **A Payoff Report (.rep) with diagnostics (optionally)**

Naïve Calibration Setup

- **Copy the data files you created from the data generator model to the “start” model folder**

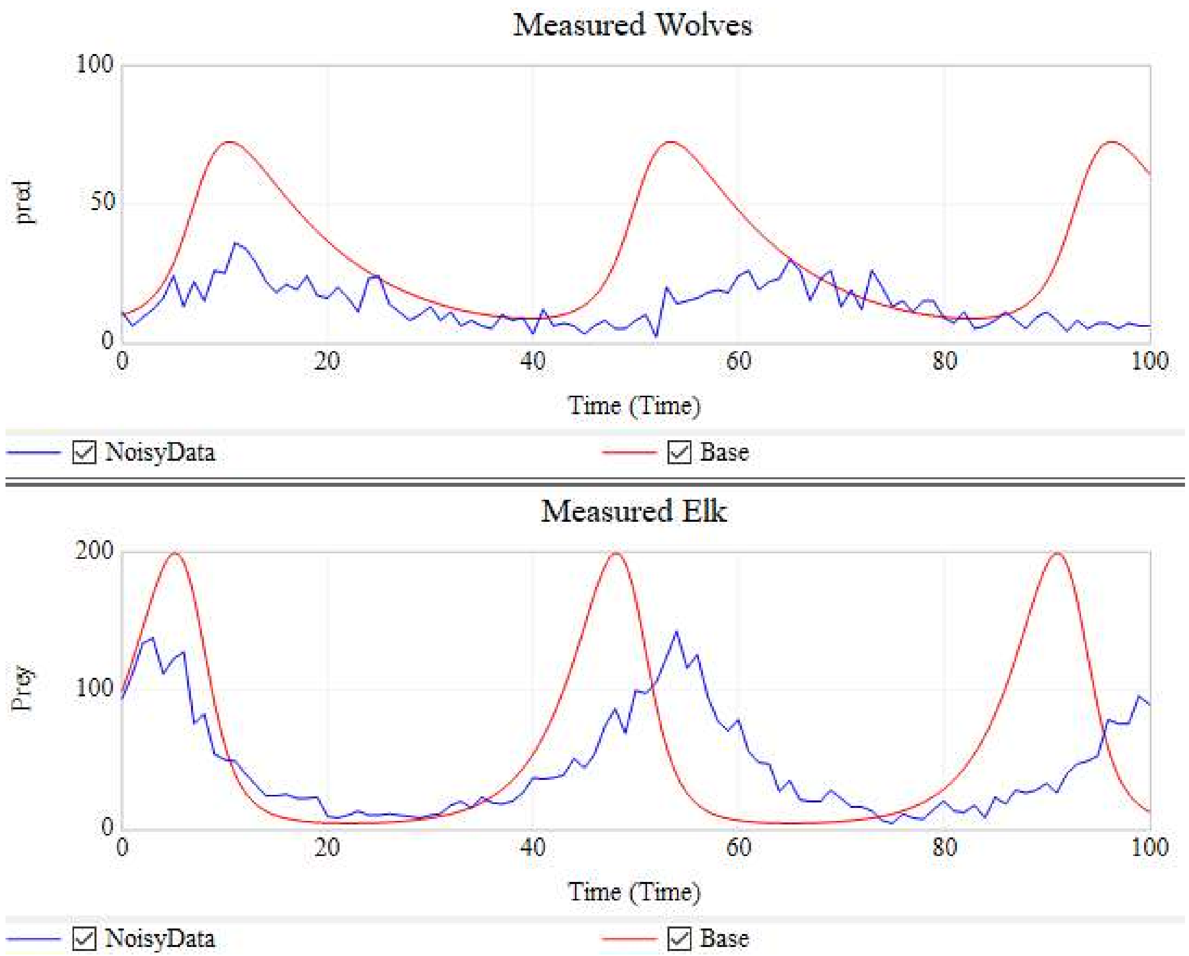


- **Open the starting point model**



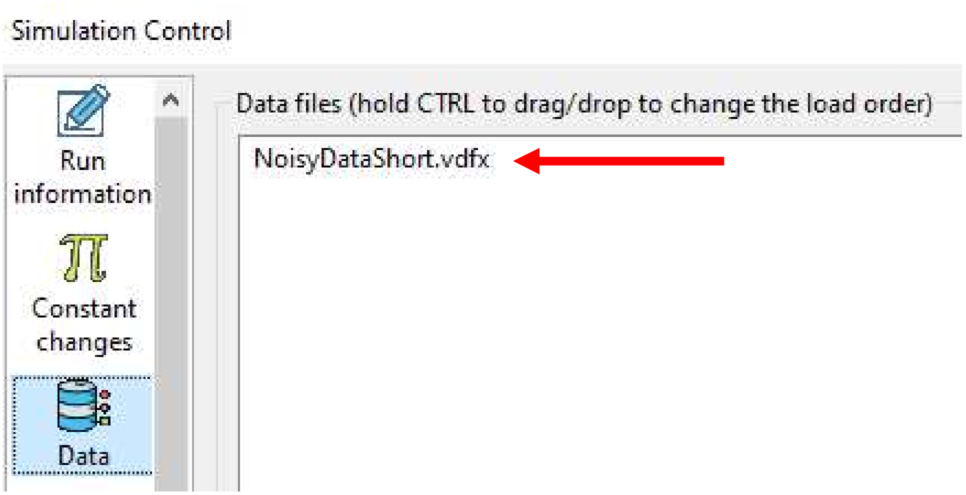
Naïve Calibration Setup

- Do a “Base” run (uncalibrated)
- Load the data from NoisyData or NoisyDataShort
- Notice how the model doesn’t fit the data well



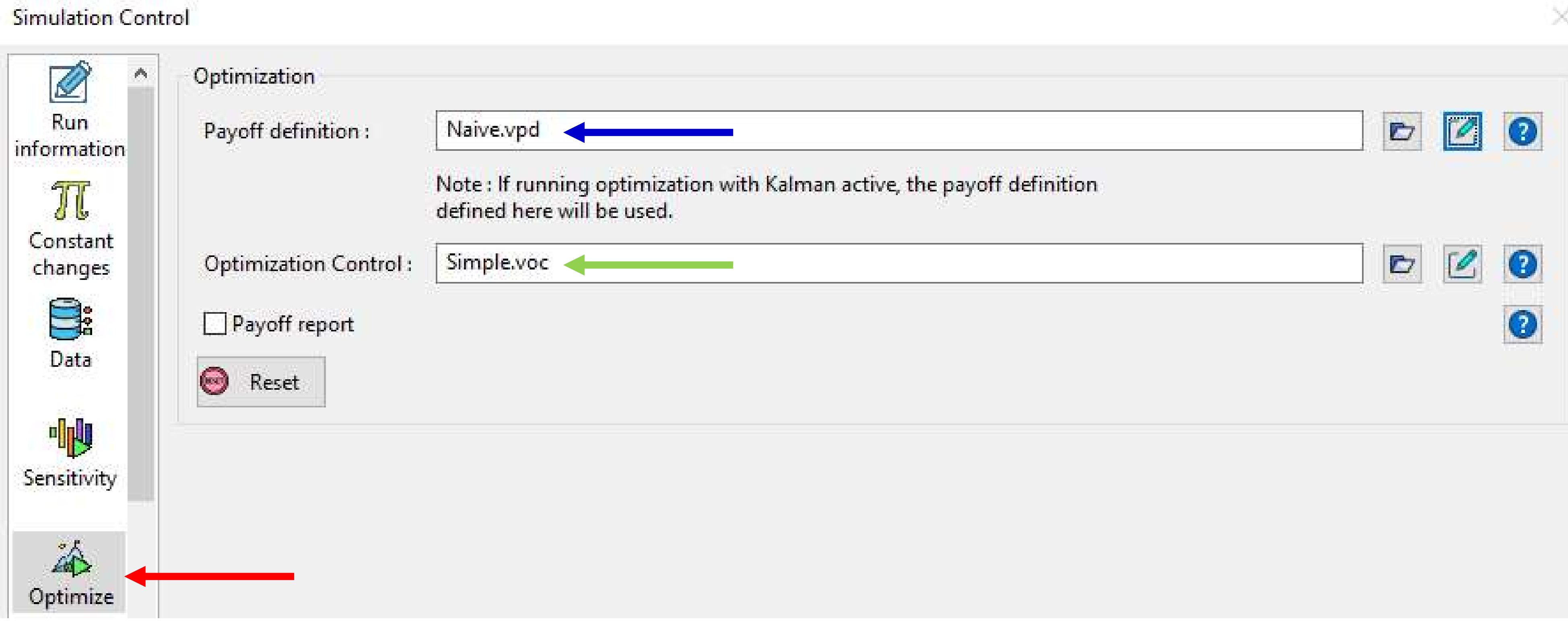
Naïve Calibration Setup

- **Change the runname ("NaiveCal.vdfx" or similar)**
- **Go to the Data tab**
  - Load comparison data (recommend the short version)

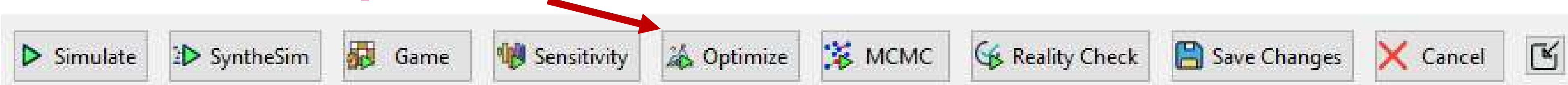


Naïve Calibration Setup (Continued)

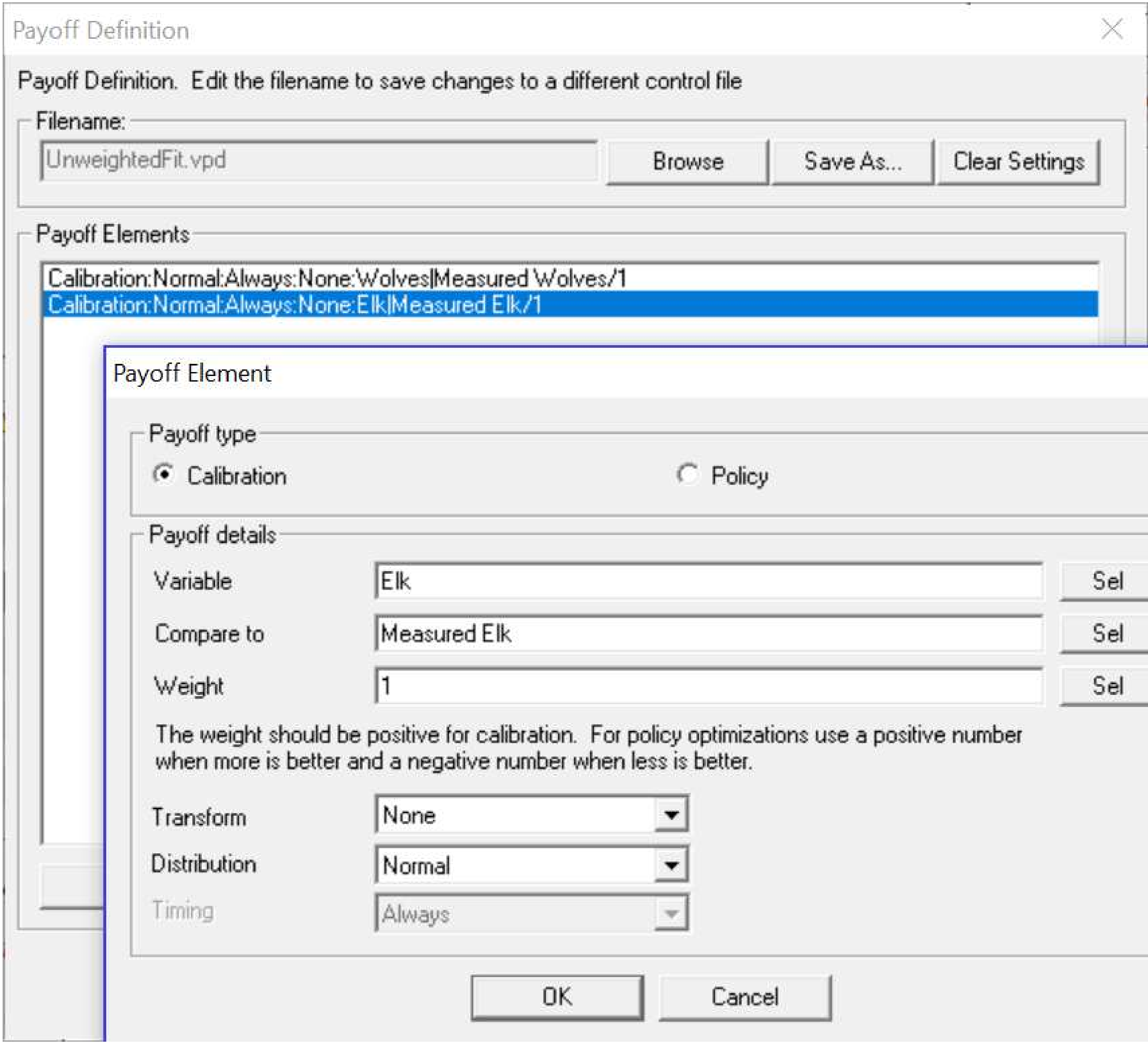
- **Change the runname ("NaiveCal.vdfx" or similar)**
- **Go to the Optimize pane**
  - Create a Payoff (.vpd)
  - Create a Control file (.voc)



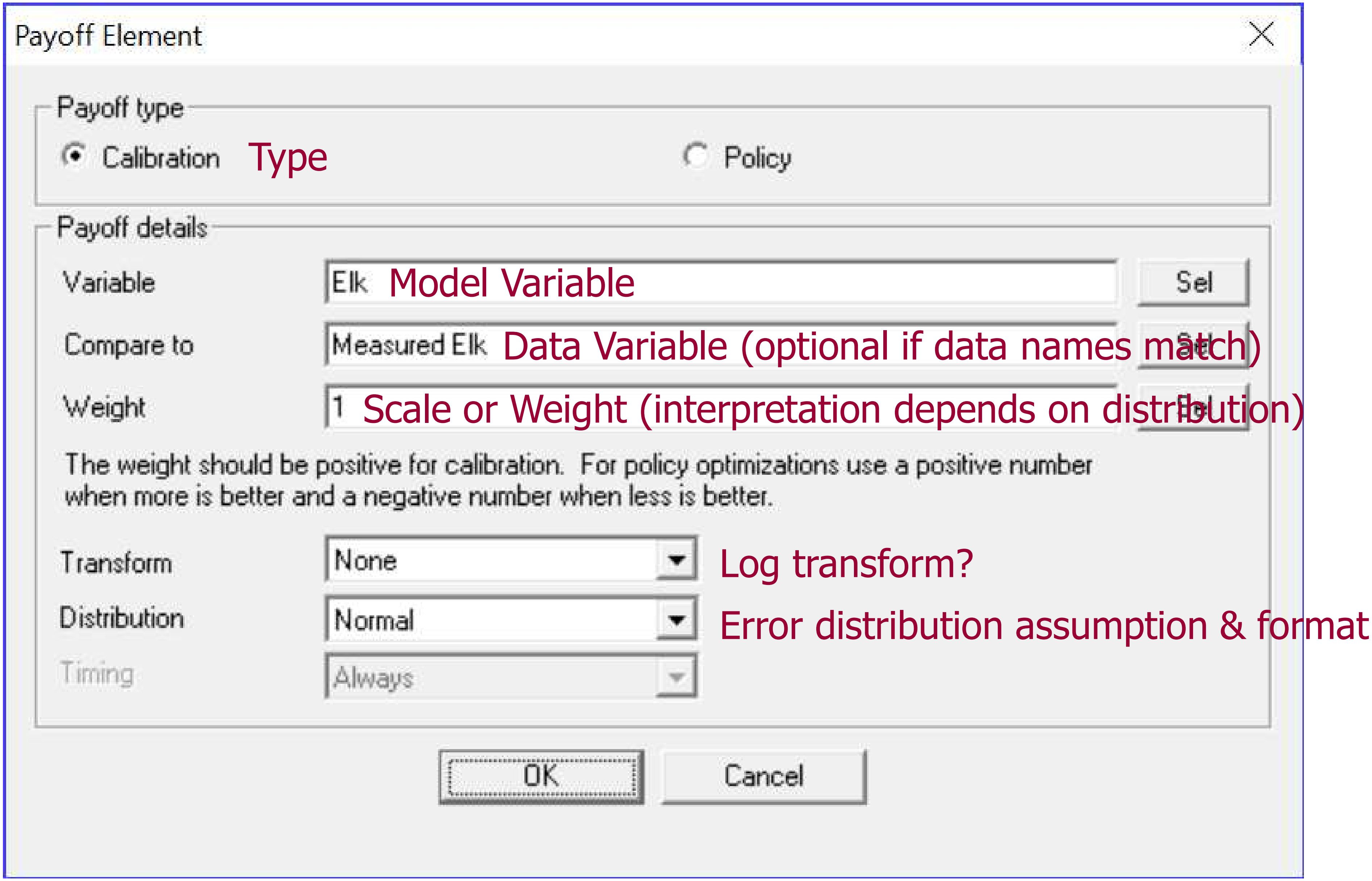
- **Hit the Optimize button**



Payoff (.vpd)


Payoff Definition
Payoff Definition. Edit the filename to save changes to a different control file
Filename:
UnweightedFit.vpd
Browse
Save As...
Clear Settings
Payoff Elements
Calibration:Normal:Always:None:Wolves|Measured Wolves/1
Calibration:Normal:Always:None:Elk|Measured Elk/1
Payoff Element
Payoff type
Calibration
Policy
Payoff details
Variable
Elk
Sel
Compare to
Measured Elk
Sel
Weight
1
Sel
The weight should be positive for calibration. For policy optimizations use a positive number when more is better and a negative number when less is better.
Transform
None
Distribution
Normal
Timing
Always
OK
Cancel

Calibration Payoff Types



The Payoff File (.vpd) as text

```
*C
```

Keyword indicating type (calibration = *C, policy = *P, etc.)

```
wolves|measured wolves/1
```

Model variable | data variable / weight or scale parameter

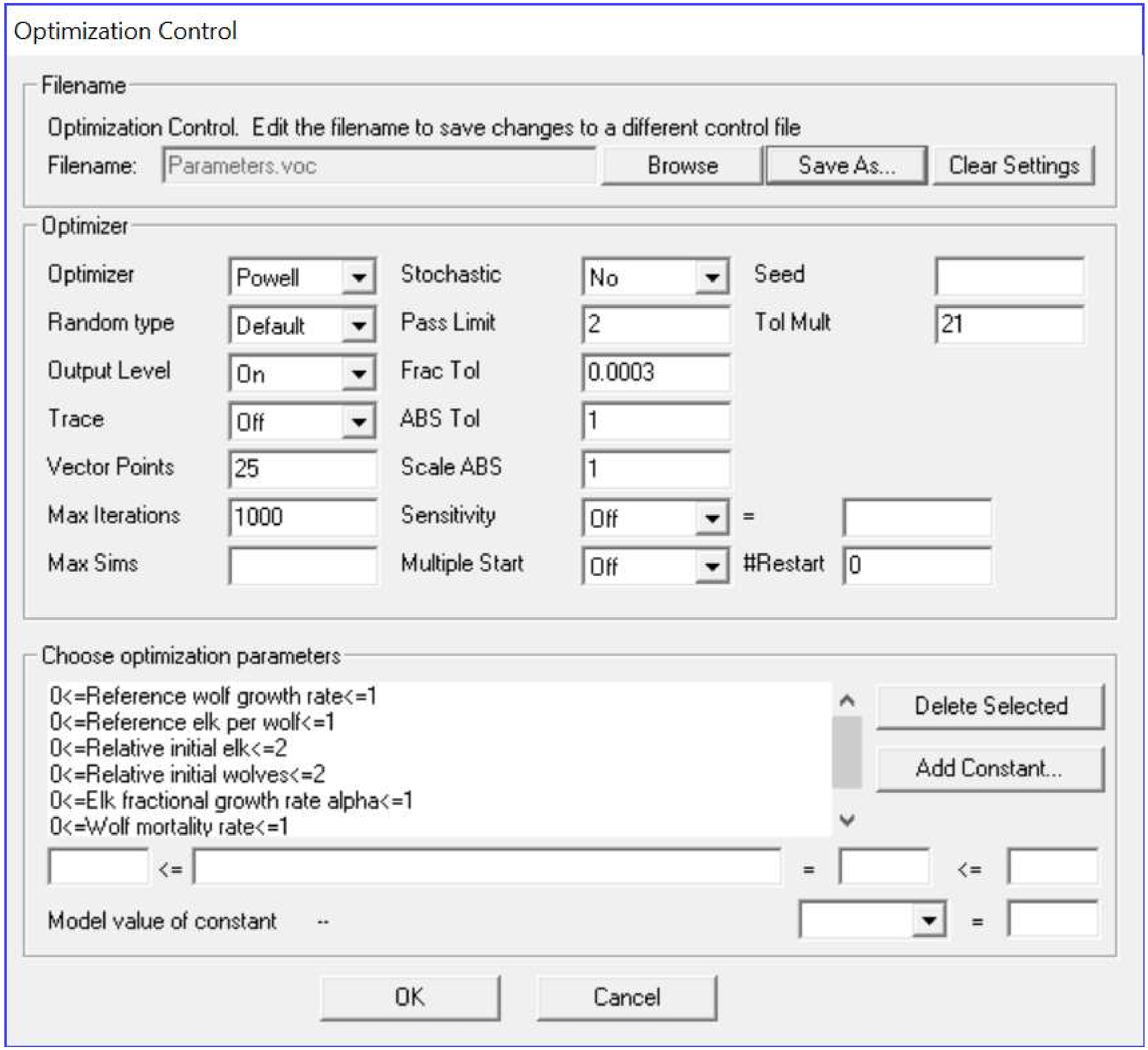
The weight can also be a variable.

Subscript ranges are OK, as long as they match.

Optimization Control File (.voc)

Method & Settings

Parameters & Bounds



The Optimization Control File as text

```
:OPTIMIZER=Powell
:SENSITIVITY=Off
:MULTIPLE_START=Off
```

<bla bla bla – algorithm control settings>

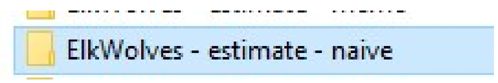
List of parameters to optimize:

```
0<=Reference wolf growth rate<=1
0<=Reference elk per wolf<=1
0<=Relative initial elk<=2
…
```

Min <= Variable Name = Initial Guess <= Max

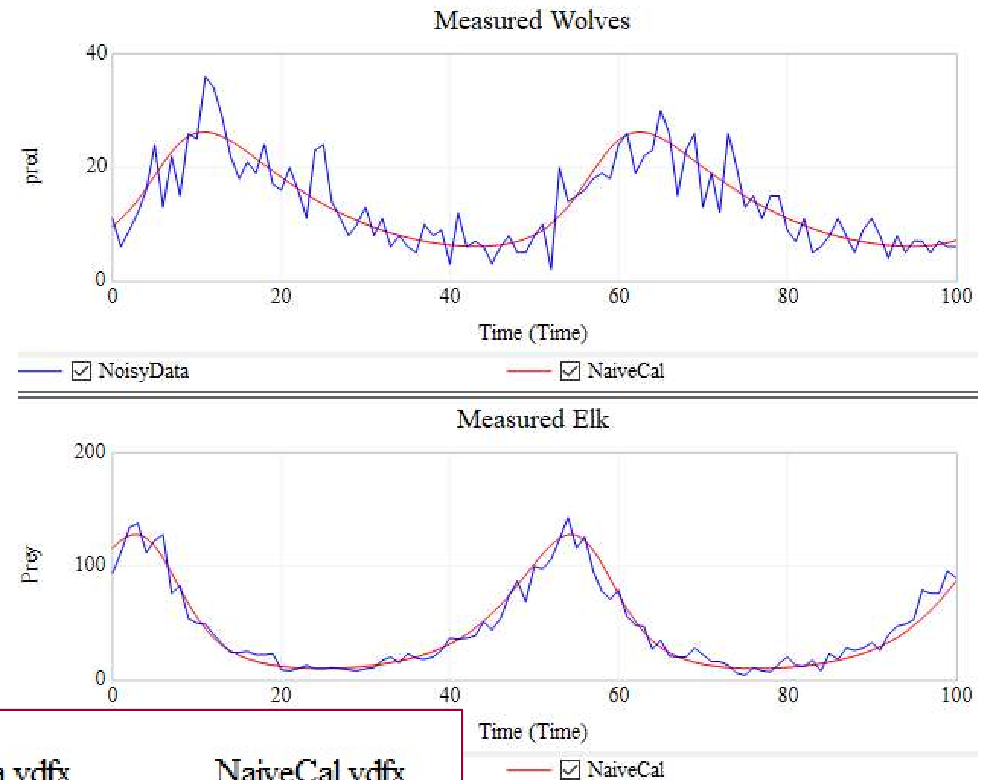
Subscript ranges are OK. Initial Guess is often omitted.

Optimize
Sample results are in



- **The model now fits the data (hopefully)**
- **It fits in the future, after the short data runs out (we probably made this too easy)**

- **Verify: use the Runs Compare tool to see if the parameters match the synthetic data model**



NoisyData.vdfx	NaiveCal.vdfx
0.22	0.22226253
0.2	0
0.2	0
0.16	0.16122551
0.18000001	0.16827042
1.08	1.1602107
0.94999999	0.96516252
0.2	0

Variations

- **Try again with ...**
  - an even shorter input data series
  - more noise in the generator
  - the 2nd order model data
  - a longer forecast horizon

VENTANA systems, inc.

Vensim® Software

Linking systems thinking to powerful dynamic models

Calibration with Vensim – Part 2

Tom Fiddaman

2022

Advanced Calibration

- **Weighting Payoff Elements**
- **Kalman Filtering**
- **Markov Chain Monte Carlo**
- **Sensitivity**

Less-Naïve Calibration

- **Weight (model-data) comparisons**

Motivation

- **To recognize varying scale and quality**
  - At different times (bigger data -> bigger error)
  - Of different measurements (#elk > #wolves, or wolf error > elk error)
- **For computation of confidence bounds**
  - A properly-weighted likelihood has a known distribution and is compatible with MCMC
- **In many cases, we can estimate the weights**

Example – Lots of elk, any wolves?

Maximum Likelihood

- **Choose the value of parameters that maximizes the likelihood of observing the data given the model**
- **This is called a Maximum Likelihood Estimator (MLE)**
- **Suppose there is more than one observation**
  - Then the likelihood is the product of the individual likelihoods for each data point
  - Working with log likelihood is easier, because ln() converts the product to a sum
- **Likelihood expresses the probability of getting the data observed from your model, not the chance that the model is right**

Likelihood Surface
Gaussian errors

- **Likelihood =** $\frac{1}{\sigma\sqrt{2\pi}}e^{-\left(\frac{model-data}{\sigma}\right)^2/2}$

- **This is the PDF of the Gaussian (Normal) distribution**

- **σ represents the standard error associated with a data point, corresponding with the weight assigned in Vensim (or its inverse)**

Log-Likelihood
Gaussian errors

- **Likelihood =** $\frac{1}{\sigma\sqrt{2\pi}}e^{-\left(\frac{model-data}{\sigma}\right)^2/2}$

- **Log(Likelihood) =**
  - LN(σ) - the bigger the σ, the lower the likelihood, as it's spread thinner
  - LN($\sqrt{2\pi}$) - this is a constant we can ignore
  - $\left(\frac{model-data}{\sigma}\right)^2/2$ - the weighted sum of squares, as in the naïve method, but for the factor /2

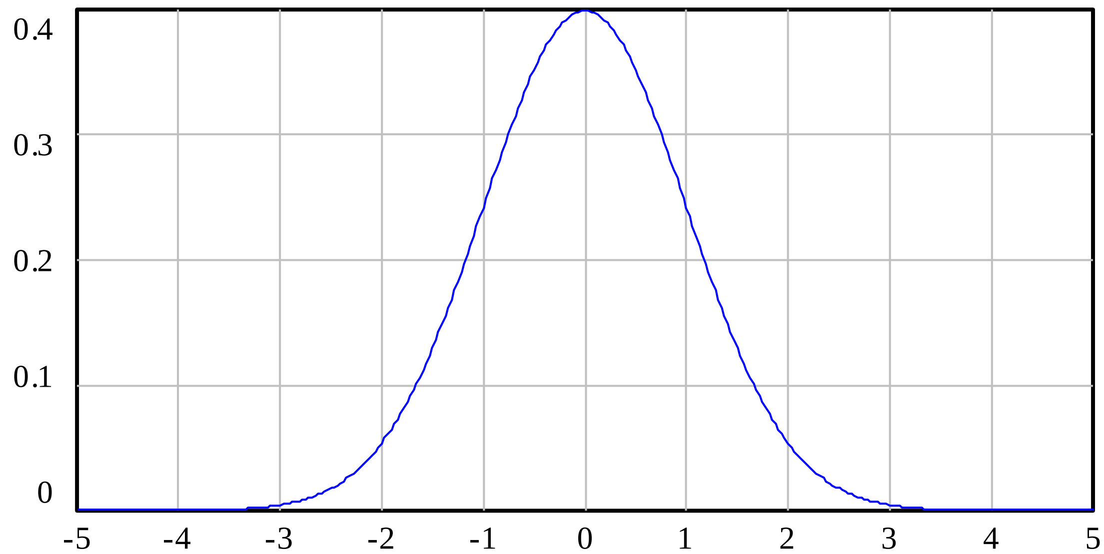
Log Likelihood
Gaussian errors

- **Likelihoods combine multiplicatively, i.e. Likelihood(A and B) = Likelihood(A)*Likelihood(B)**
- **Log likelihoods therefore sum, LN(Likelihood(A and B)) = LN(Likelihood(A)) + LN(Likelihood(B))**

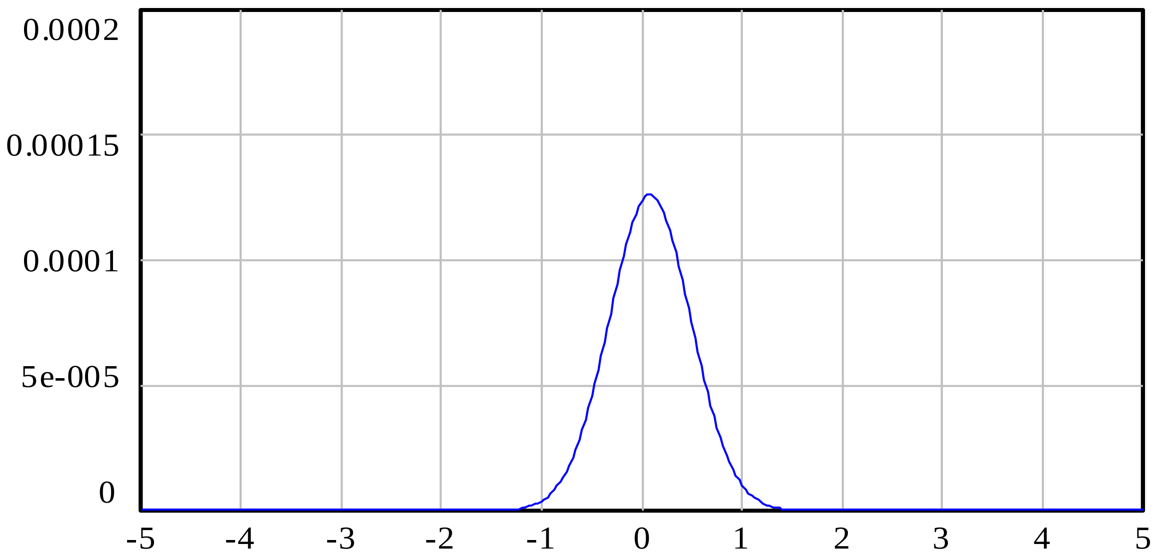
- $(\frac{model-data}{\sigma})^2$ **is** $(\frac{\text{error}}{\sigma})^2$ **so if we've guessed right about** σ**, we expect this to have magnitude ~1.**
- **For multiple data points, we expect the weighted sum of squares to have magnitude of the number of data points, and have a Chi-squared distribution.**
- **Therefore a properly-weighted payoff should have a magnitude of N or N/2 (depending on the method choice)**

Adding data shrinks the likelihood peak

One Point



Several Points



Error Distribution Assumptions

- **N – Normal (simplest – used first for naïve estimate)**
  - Payoff is the sum of $(\text{model-data})^2$*weight
  - Weight = 1/(standard error of measurement)
  - Proportional to 2*log likelihood
  - You can't estimate the weight as a parameter
- **G – Gaussian (often best choice)**
  - Sum of $((\text{model-data})/\text{StdDev})^2/2$ - LN(StdDev)
  - This is a log likelihood (up to a constant multiplier) and can be used to estimate the StdDev
- **K – Kalman**
  - Same as Gaussian, but specified with Variance instead of StdDev (primarily for use with the Kalman filter)

Error Distribution Assumptions 2

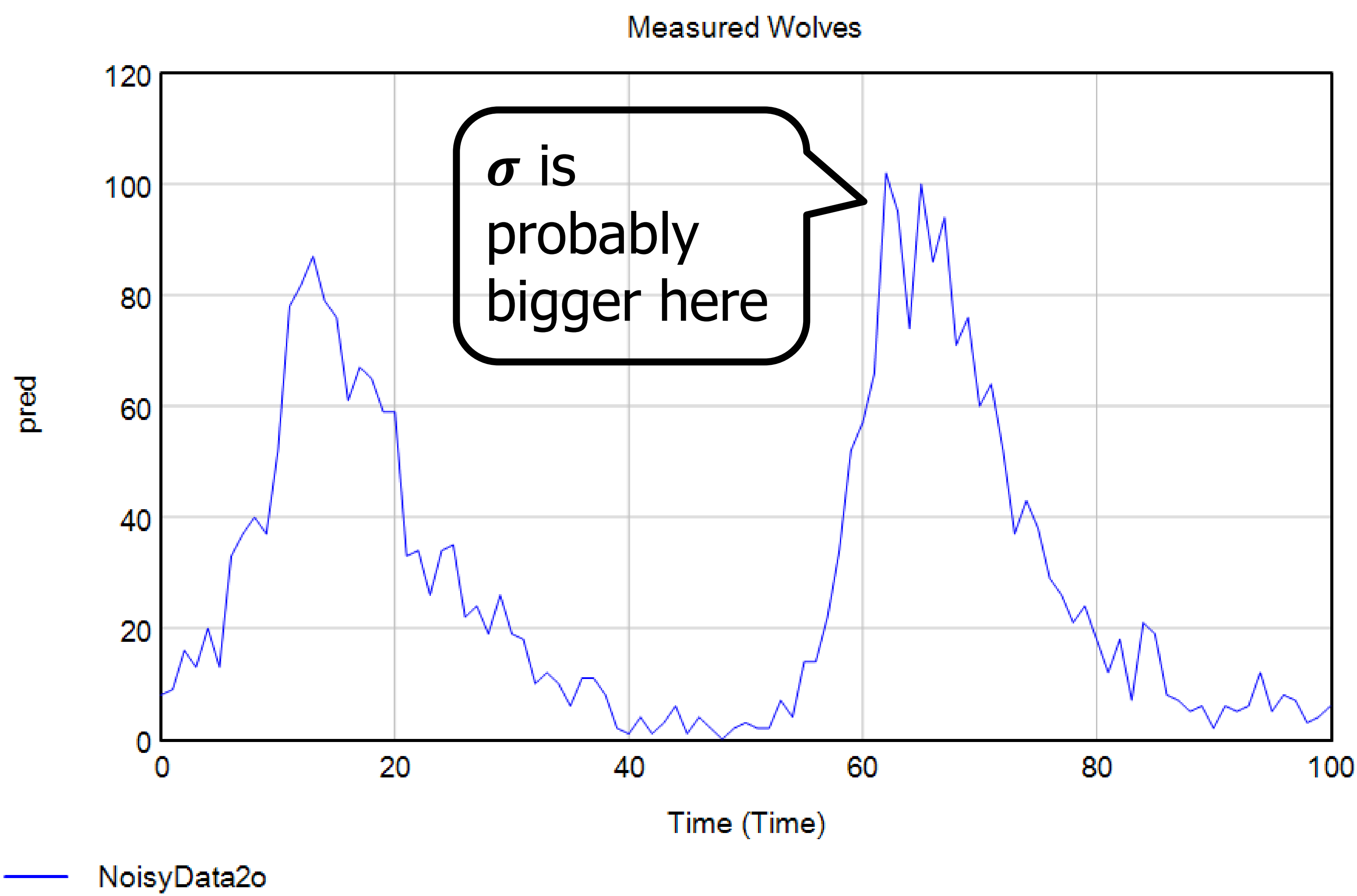
- **R – Robust**
  - Sum of ABS((model-data) /AbsErr) - LN(AbsErr)
  - AbsErr scale parameter is a median absolute deviation rather than standard deviation
  - This is a log likelihood (up to a constant multiplier) and can be used to estimate the AbsErr
  - Not as efficient as Gaussian, but resistant to contaminated data
- **(Others – Robust/Huber, Poisson, etc.)**

- **For most purposes (not COVID!), use Normal, Gaussian, Kalman or Robust**
- **Normal, Gaussian, Kalman differ only in interpretation of the weight**

How do you determine σ?

- **Guess:**
  - "plus or minus x%"
  - Standard deviation of the data (if stationary)
- **Iterate:**
  - Run the model
  - Look at the payoff or the residuals
  - Adjust the error toward what you observe
- **Estimate:**
  - Include the error or weight as an optimization control parameter
  - Requires extra terms in the payoff
  - Likelihood $= \frac{1}{\sigma\sqrt{2\pi}} e^{-\left(\frac{model-data}{\sigma}\right)^2 / 2}$

Scale Variation

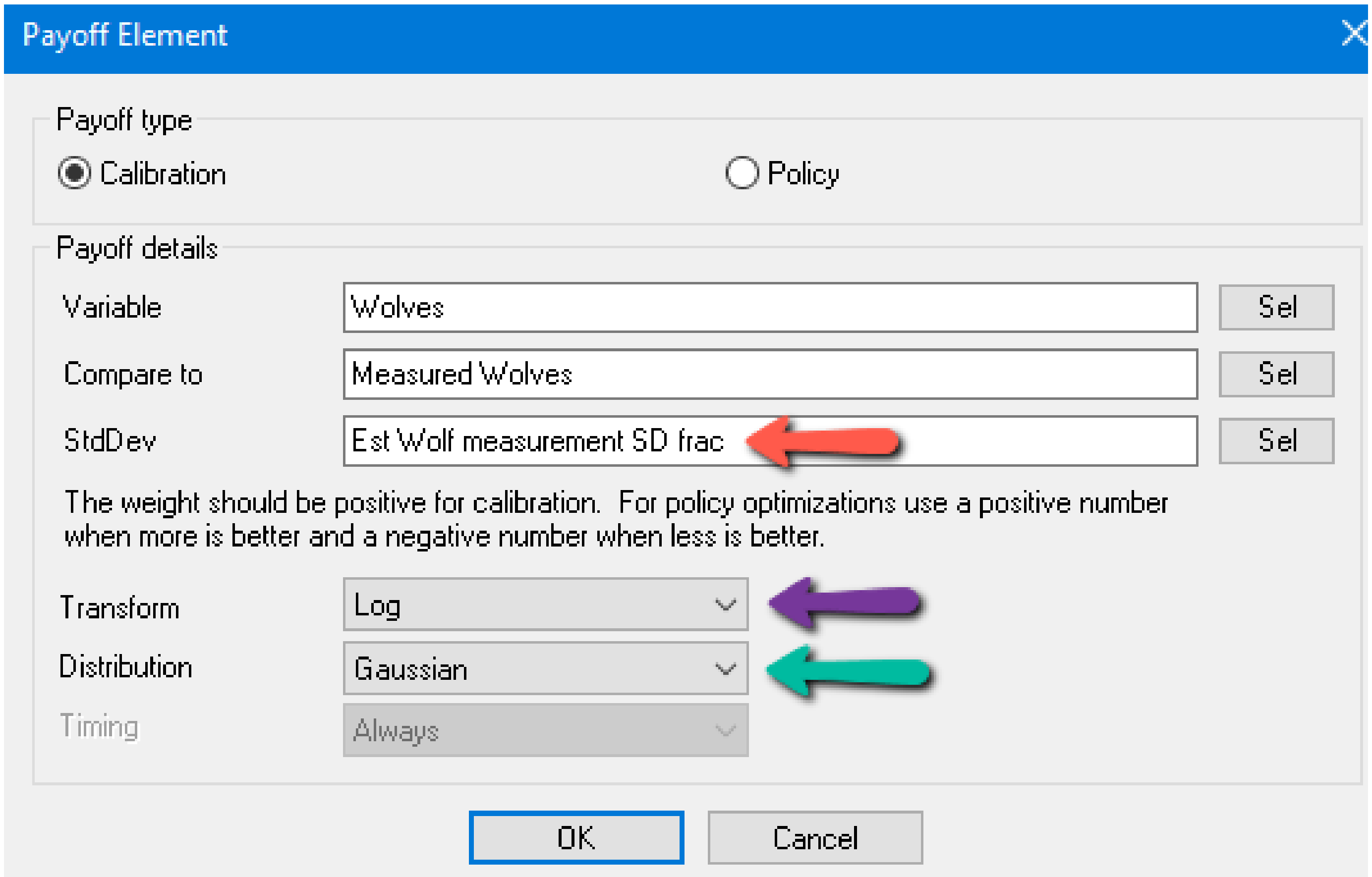


Response to Scale Variation

- Log-transform the data
- Then σ represents the fractional error, rather than the absolute error

- This doesn't work if the data includes 0, but there are other quasi-log transformations that could be used
- It also doesn't work for data with both negative and positive values, for which an absolute error makes more sense

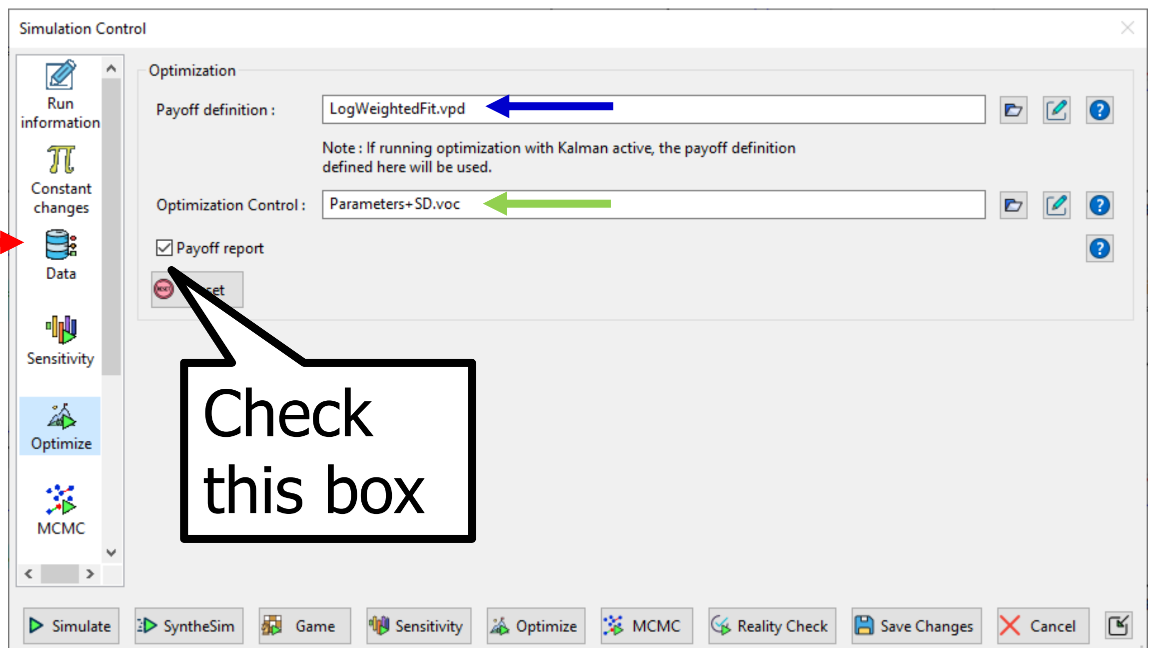
There’s an option for that …
Policy Payoff Types



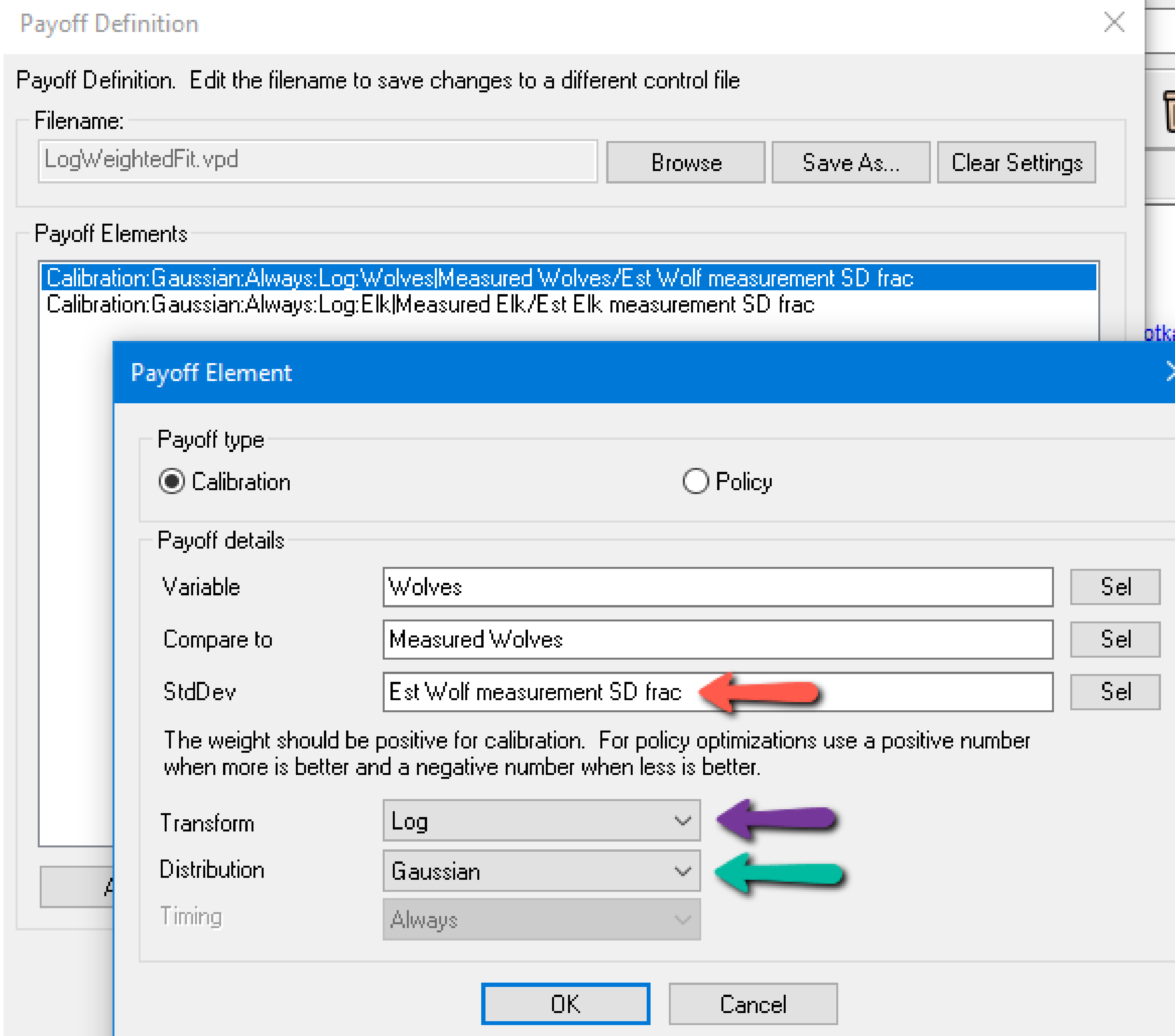
Weighted Calibration Setup

ElkWolves - estimate - weighted

- **Go to the Advanced tab**
  - Load comparison data (recommend NoisyDataShort.cin)
  - Create a Payoff (.vpd) – different weighting
  - Create a Control file (.voc) – adds error terms
- **Hit the Optimize button**



Payoff (.vpd)

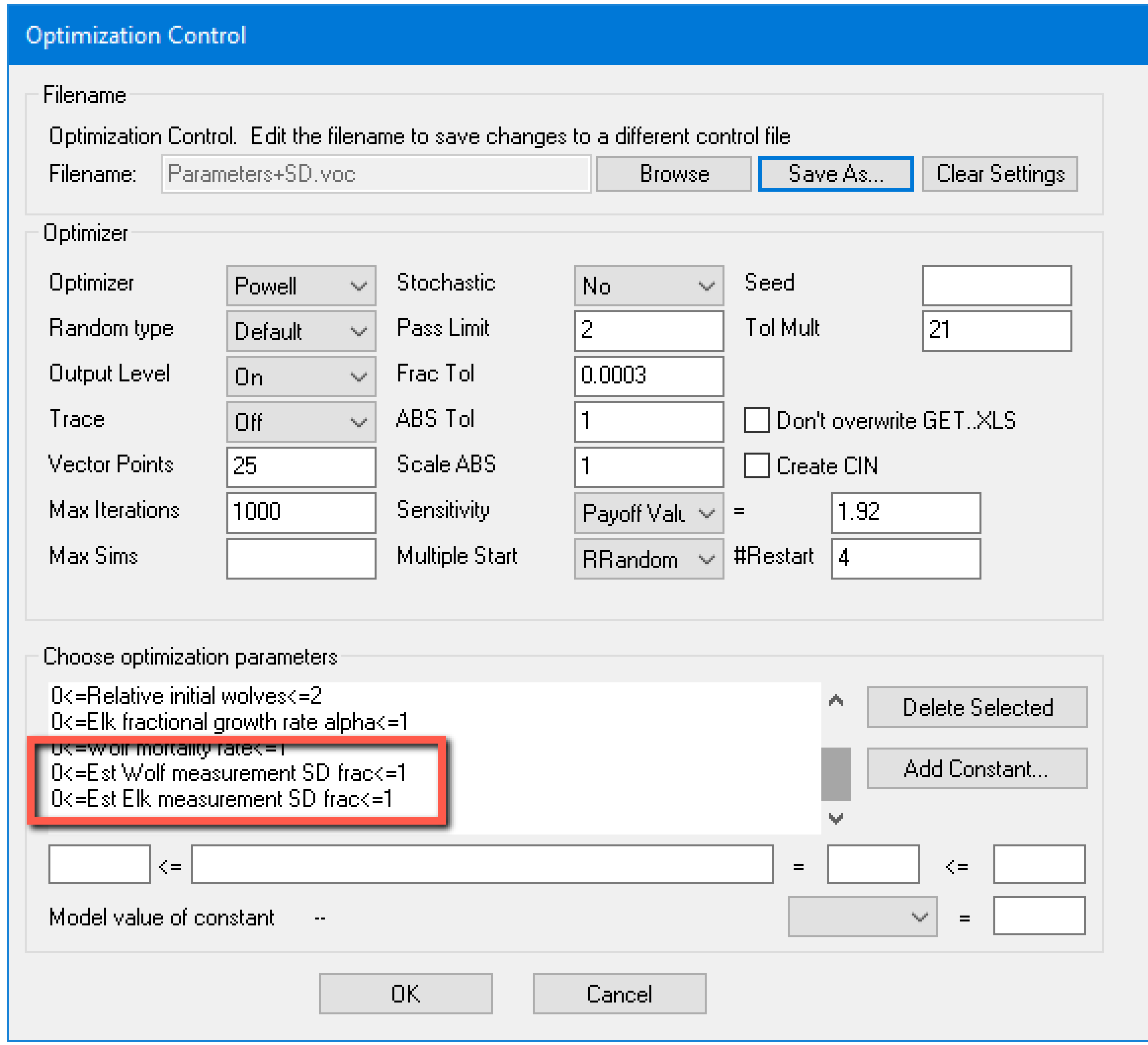


Payoff Definition
Payoff Definition. Edit the filename to save changes to a different control file
Filename:
LogWeightedFit.vpd
Browse
Save As...
Clear Settings
Payoff Elements
Calibration:Gaussian:Always:Log:Wolves|Measured Wolves/Est Wolf measurement SD frac
Calibration:Gaussian:Always:Log:Elk|Measured Elk/Est Elk measurement SD frac
Payoff Element
Payoff type
Calibration
Policy
Payoff details
Variable
Wolves
Sel
Compare to
Measured Wolves
Sel
StdDev
Est Wolf measurement SD frac
Sel
The weight should be positive for calibration. For policy optimizations use a positive number when more is better and a negative number when less is better.
Transform
Log
Distribution
Gaussian
Timing
Always
OK
Cancel

Optimization Control File (.voc)

Method & Settings
(no change)

Parameters & Bounds
(adds error terms)



Payoff Report

- **Open the _runname_.rep file (a text editor is OK, but Excel is better for viewing)**
- **Contents, for each data series:**
  - Contribution to payoff
  - Source of data, # of points
  - R^2
  - Durbin Watson & Autocorrelation
  - Theil statistics
    - MSE = mean squared error, Um = unequal means, Us = unequal variance, Uc = unequal covariance
  - MAE, MAPE, MAEoM

Addressing Pitfalls – Kalman filtering

- State dependent noise
- Sample size
- Data quality
- Autocorrelated errors
- Error covariance
- Measurement error
- State estimation
- Endogeneity

The General Problem

- If the state of the model has drifted away from the state of the world, the model's incremental responses are likely to be wrong

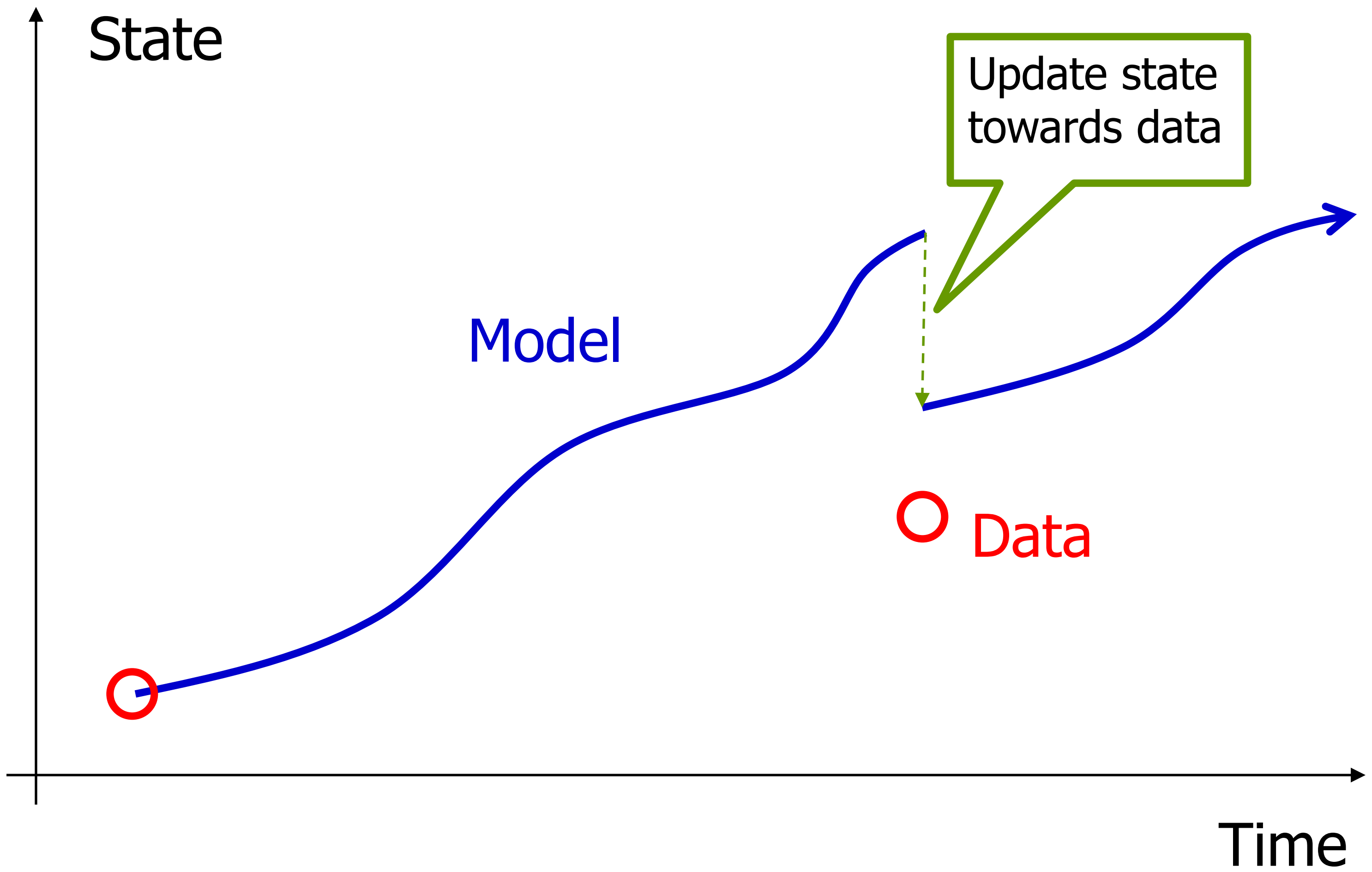
- Ordinary Least Squares on first differences essentially assumes that the data is always right
- Ordinary simulations assume that the model is always right

- Ideal: blend the apparent state in the data with the model's estimate of system state (which includes information from prior data)

Example: GPS mapping

- **The observer has six states:**
  - Position X, Y, Z (lat, lon, altitude)
  - Velocity dX, dY, dZ
- **The device takes intermittent noisy measurements of position only**
- **A simple approach to noise is to smooth successive position estimates, but that introduces a lag – we can do better with a model**
- **From physics: Position = Integral(Velocity)**
- **Strategy:**
  - Maintain estimates of position and velocity states
  - Integrate velocity to predict position changes
  - Update towards the measurements as they arrive

Kalman Filtering



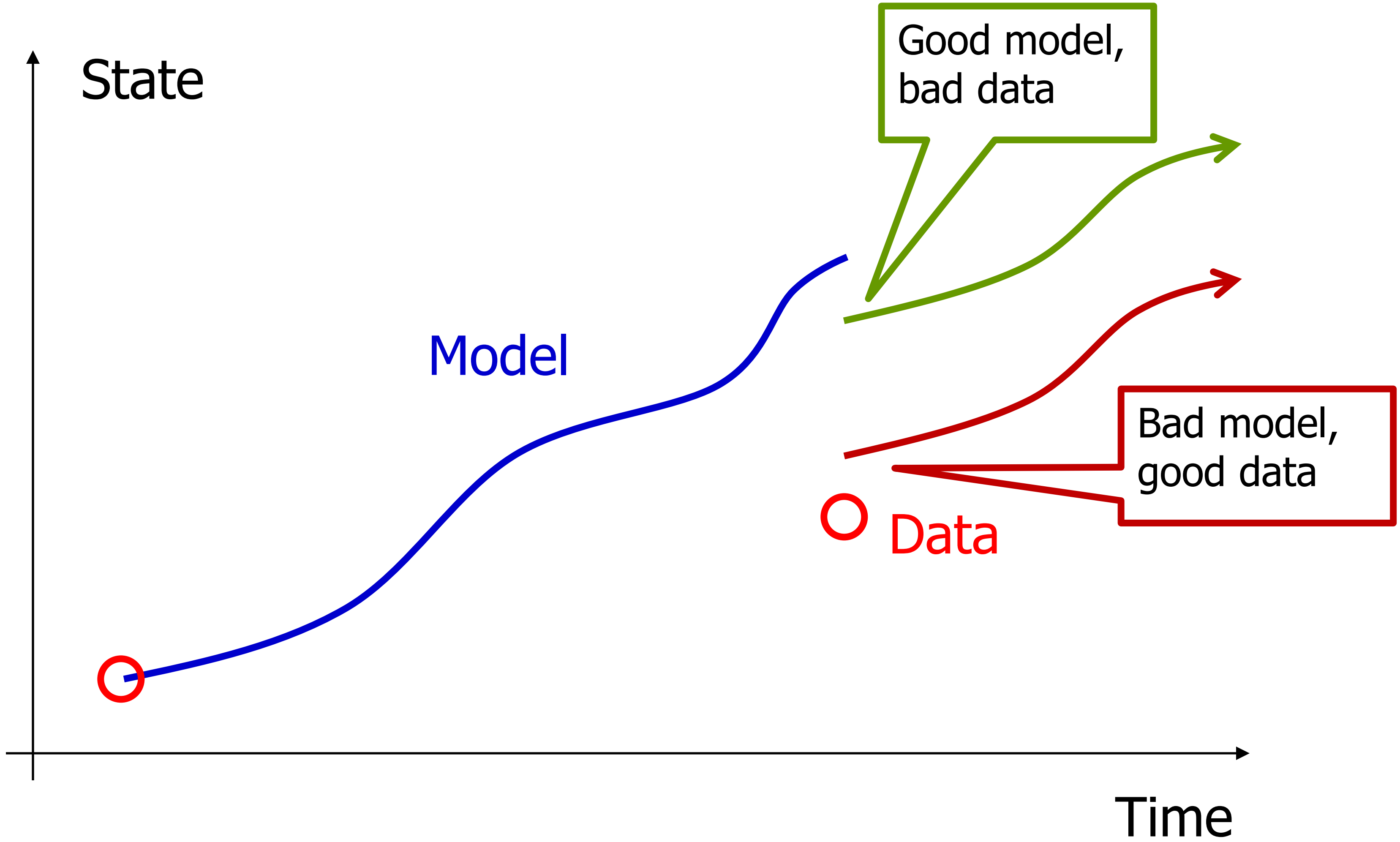
How far to update?

- **Consider:**
  - How reliable is the data?
  - How reliable is the model up to that point?

- **Bayesian update (assuming Gaussian errors):**
  - New state = variance-weighted combination of model and data = $(Model/Var_{model} + Data/Var_{data})/(1/Var_{model}+1/Var_{data})$
  - Update variance similarly

- **Complications**
  - Need to consider covariance (track N_{states}^2)
  - Data might not measure states directly (need linear algebra)
  - Non-Gaussian errors & outliers

Kalman Filtering



Is the forecast in the confidence bounds?

Why Confidence Bounds? Perspectives

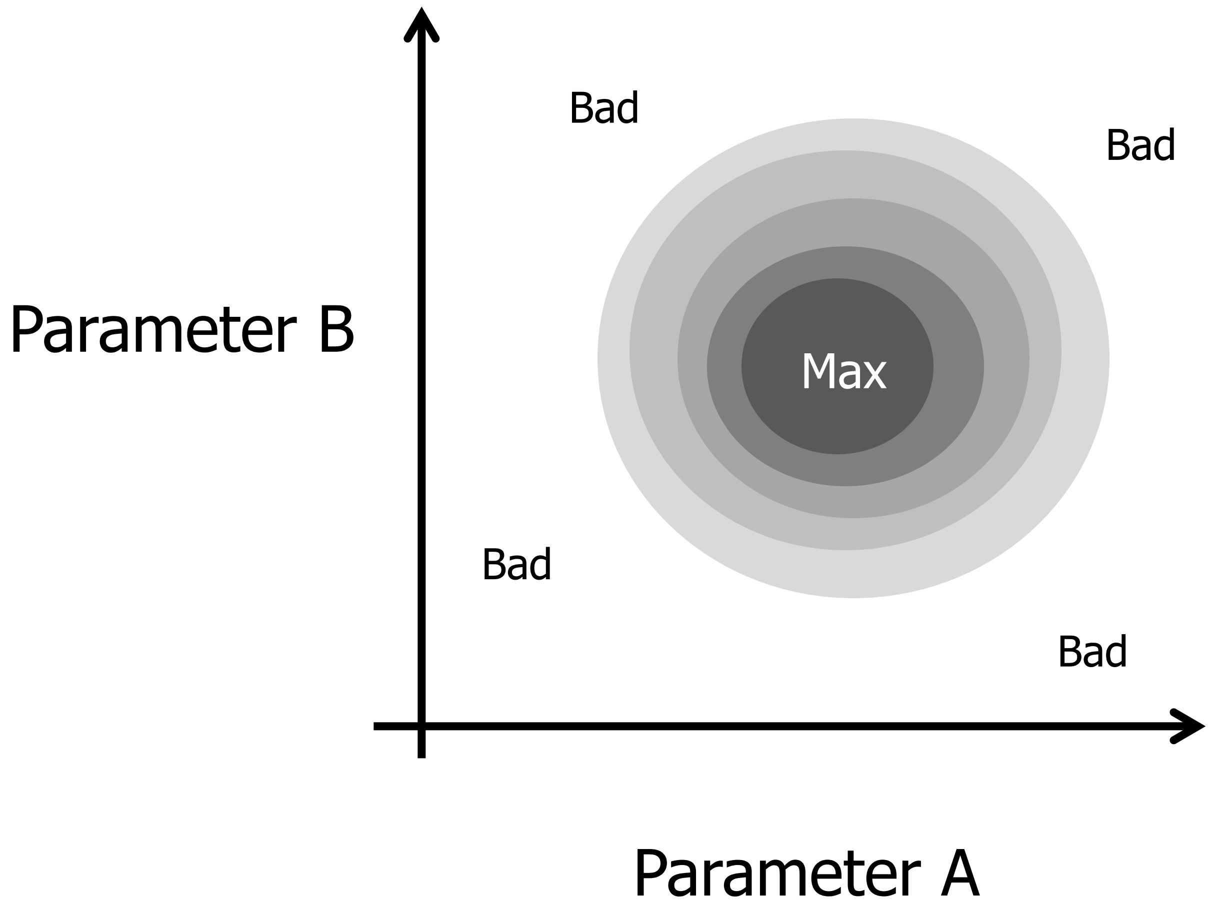
- **Statistical**
  - Is an effect significantly different from zero?
- **Practical**
  - What does uncertainty imply for policy?
  - What data might narrow the bounds?

Several Paths to Confidence Bounds

- **Old way**
  - Optimize to find the best fit to data
  - Explore the payoff surface around the maximum

- **New ways**
  - Bootstrapping (draw samples from the data)
  - Markov Chain Monte Carlo (MCMC)

Multidimensional Likelihood

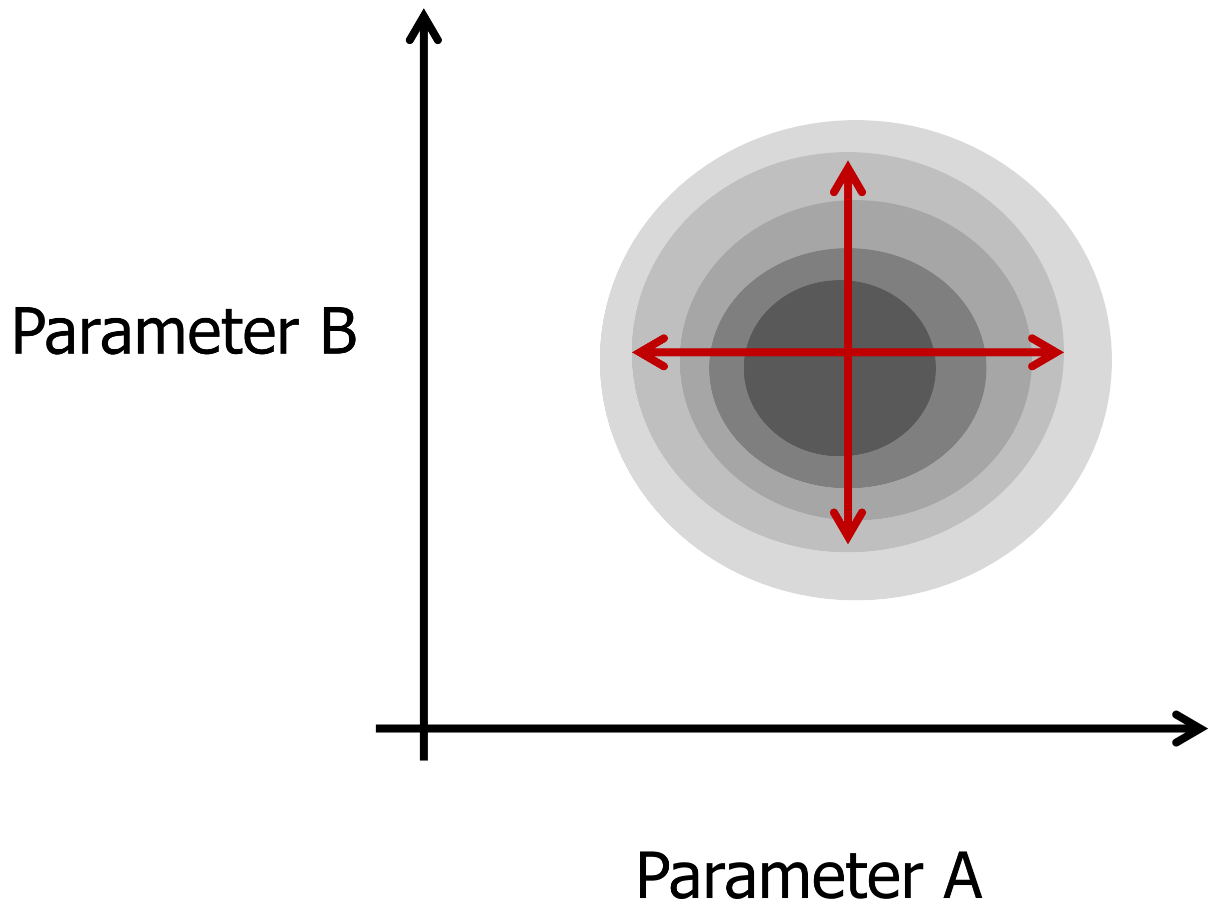


Confidence Bounds & Likelihood

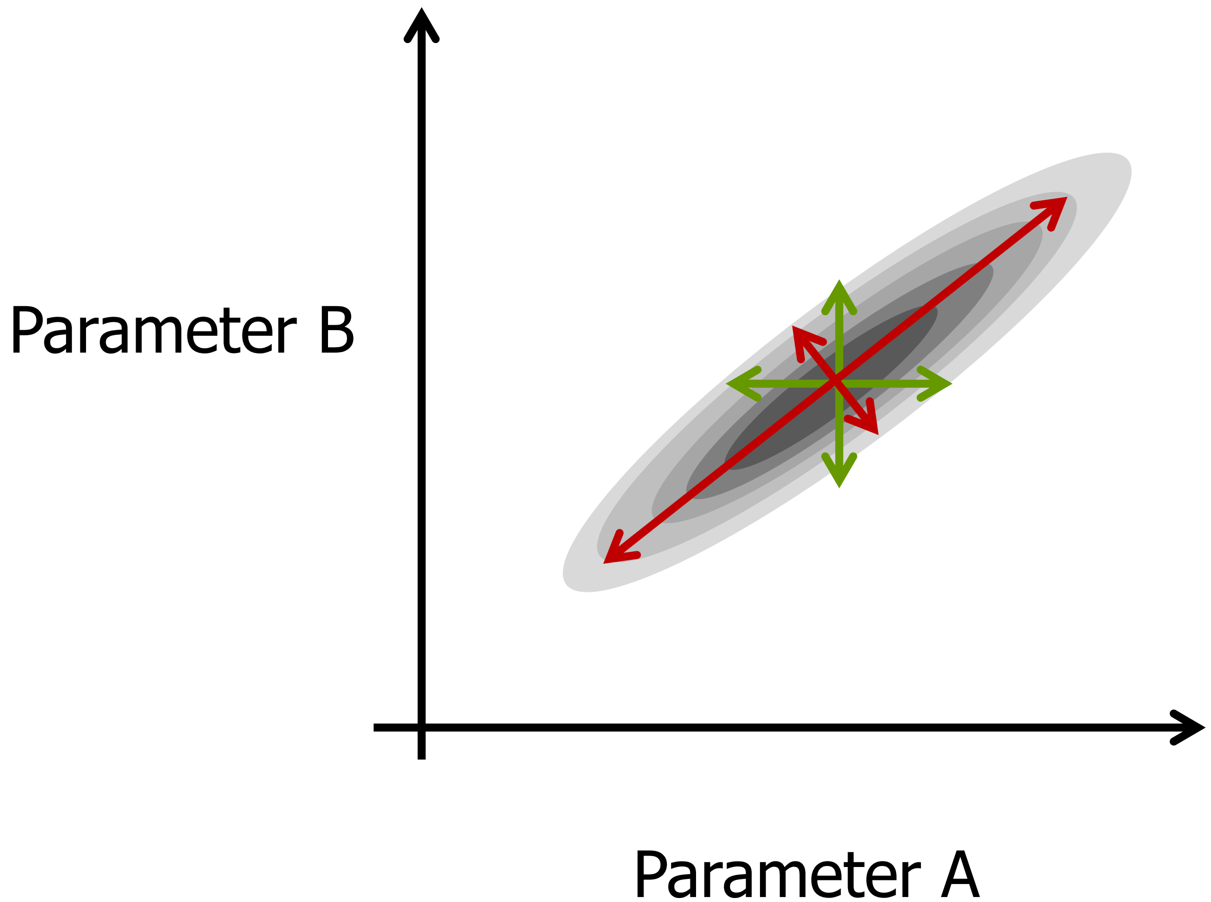
- **Gaussian Likelihood =** $\frac{1}{\sigma\sqrt{2\pi}} e^{-\left(\frac{model-data}{\sigma}\right)^2 / 2}$
- **Log Likelihood ≈** $-\ln\sigma - \left(\frac{model-data}{\sigma}\right)^2 / 2$ (leaving out invariant terms)
- **A weighted log-likelihood calibration payoff is a sum of squares; 2*Log(likelihood/best likelihood) is distributed Chi-squared with one degree of freedom**
- **The expected value is the number of data points**
- **Varying the payoff by the ChiSq critical value at 95% yields a 95% confidence bound**
  - If your payoff uses the "Normal" distribution setting, 3.84
  - If you use "Gaussian" (preferred), 1.92 (=3.84/2)
  - (Difference is due to presence or absence of the /2 factor)

Standard Vensim payoff value sensitivity

- **Test the payoff surface in the direction of each parameter independently**



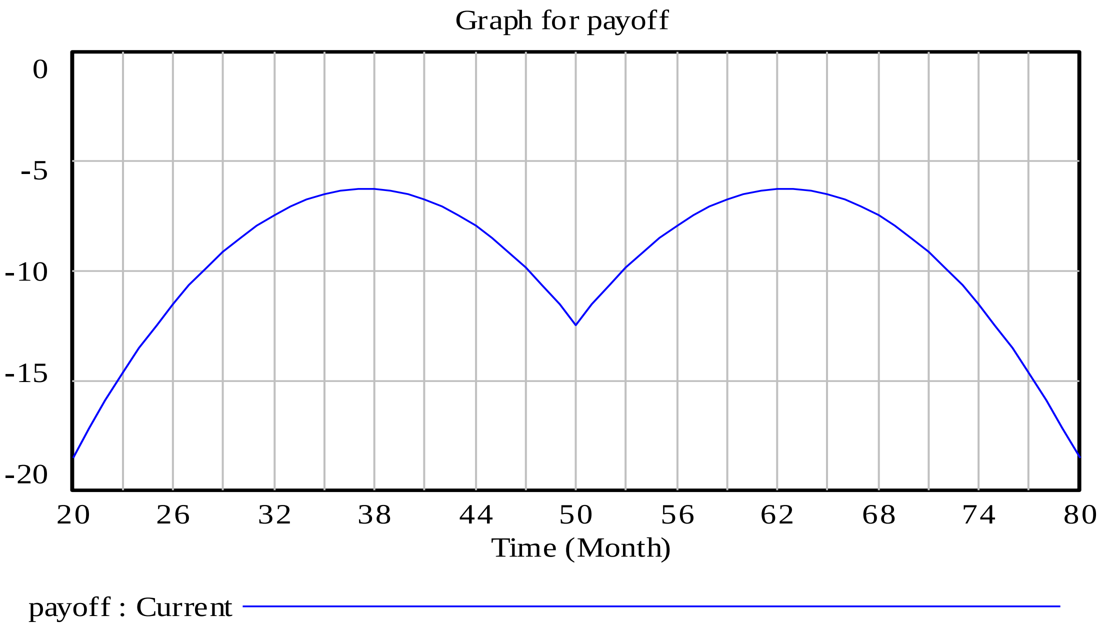
Misses off-axis ellipsoids!



- **Even harder if the likelihood surface is shaped like a banana, or a snake, or a bag of 10-dimensional jellybeans…**

Unimodality, Smoothness

- **If not, the confidence bounds can be misleading**

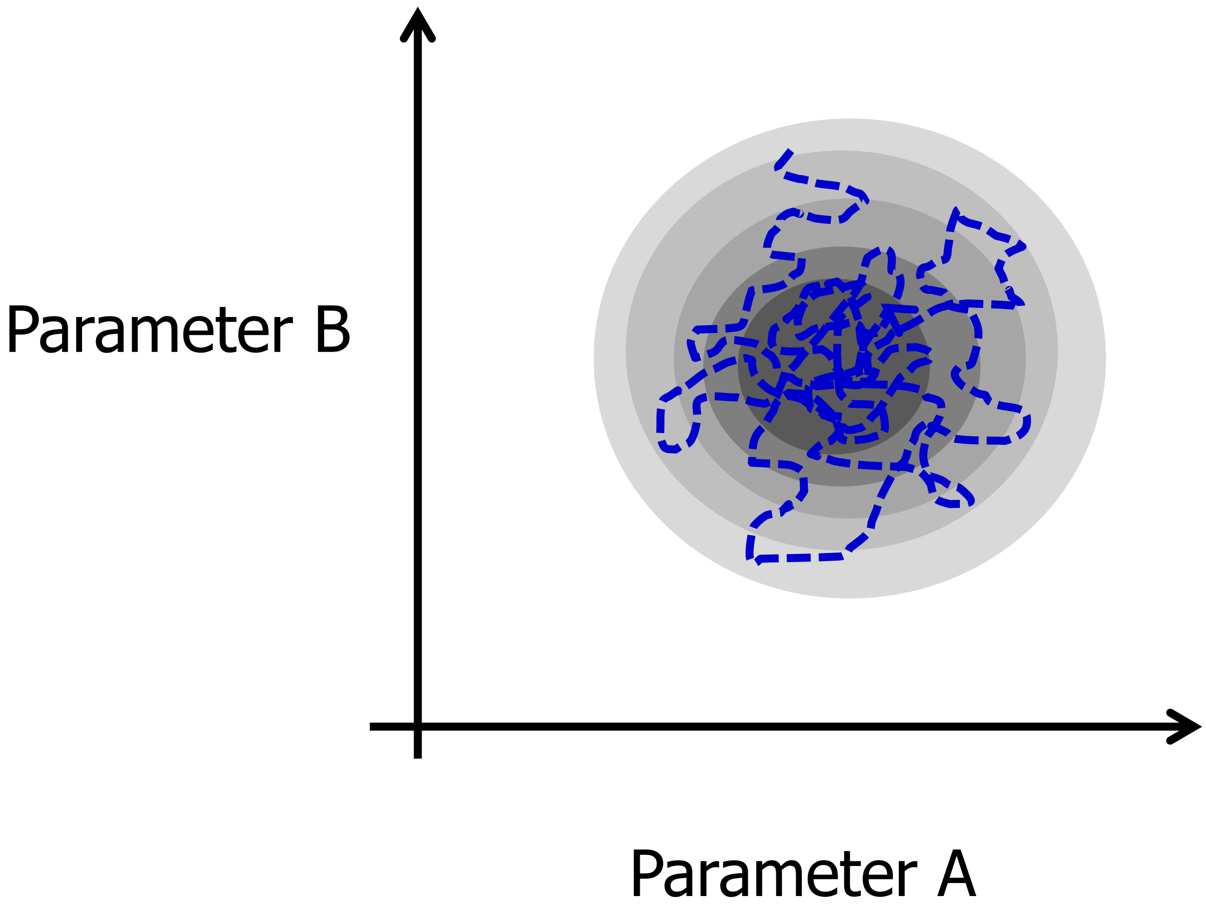


Alternate Approach to Estimation
Markov Chain Monte Carlo (MCMC)

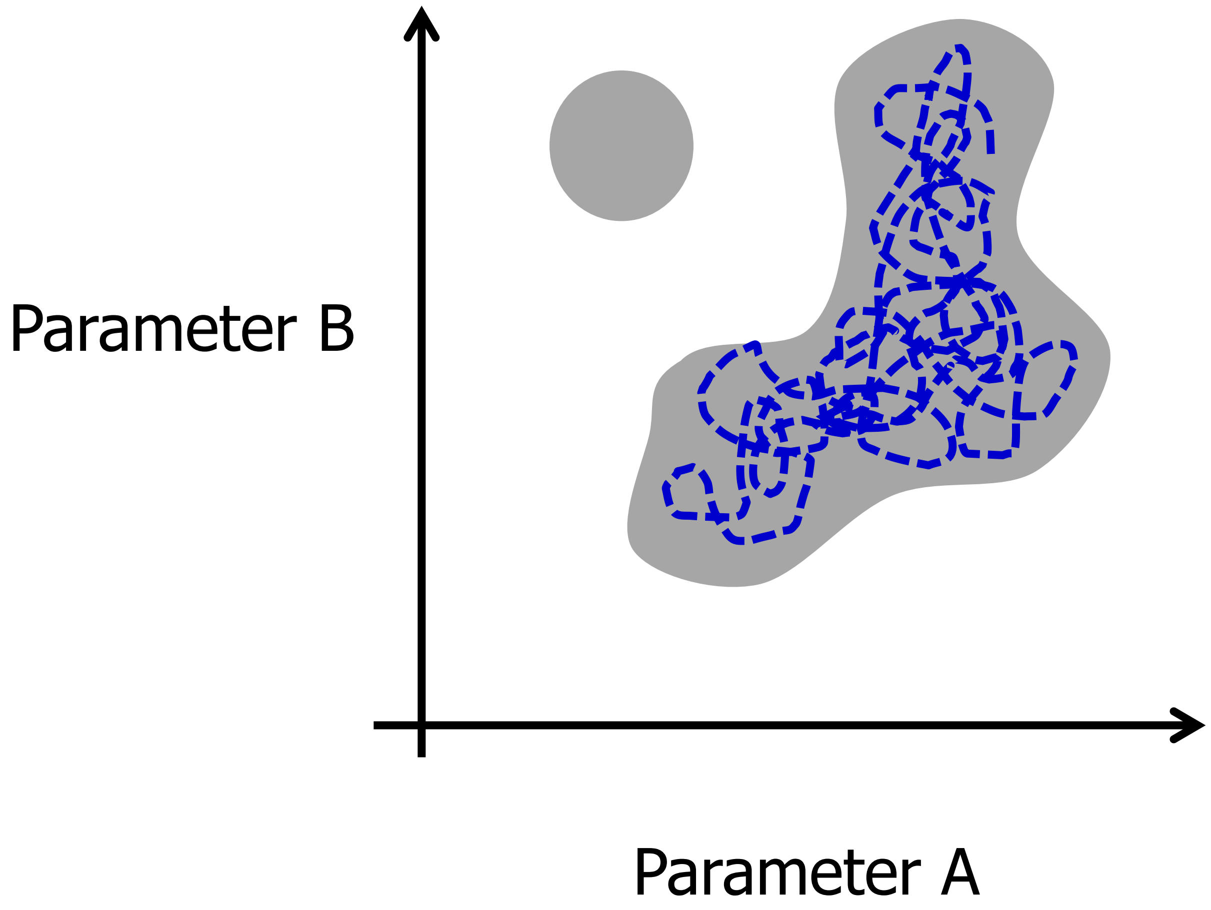
- **Perform a random walk over the payoff surface, with moves chosen according to point likelihoods**
- **Stationary distribution of the Markov process reflects likelihood surface**

- **Problem: determining scale of proposed jumps**
- **Solution: Differential Evolution (run multiple Markov chains and recombine from population to propose jumps)**

MCMC



MCMC



Procedure

- **Payoff**
  - We want the input to be (reasonably close to) a log-likelihood, so use the same kind of properly-weighted payoff we already developed

- **Control file**
  - We can use the same parameter set

  - Change Optimizer to MCMC
  - Possibly set other options

The Optimization Control File

```
:OPTIMIZER=MCMC
:MCLIMIT=5000
```
total number of runs

```
:MCBURNIN=4000
```
runs to discard as warmup

… etc. See Help system for details.

List of parameters to optimize:

```
0<=Reference wolf growth rate<=1
0<=Reference elk per wolf<=1
0<=Relative initial elk<=2
…
```

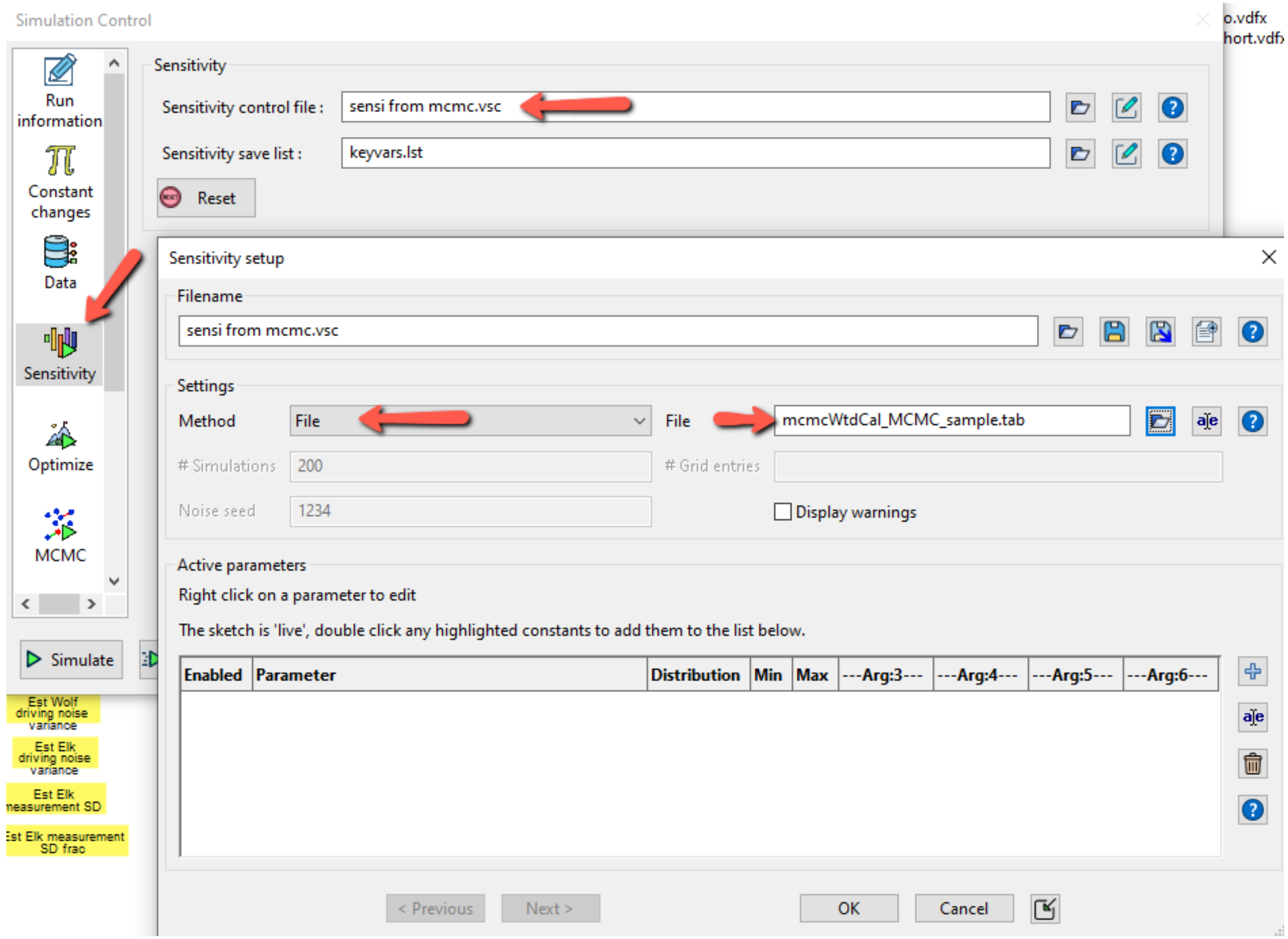
(same as before)

MCMC – the Output

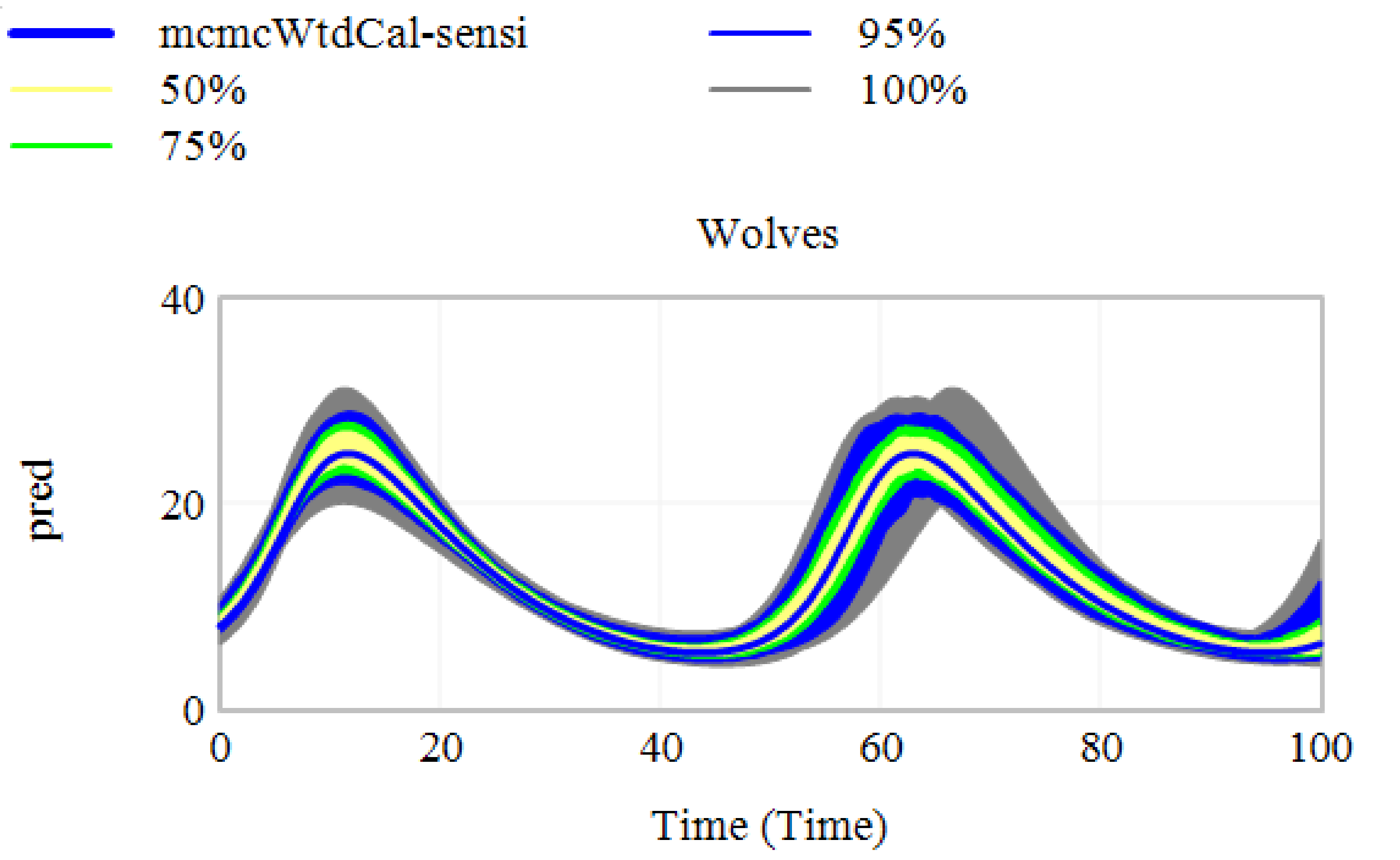
- **Three parts:**
  - _runname_MCMC_sample.tab: A sample of points representing the likelihood surface - the sample's statistics give you confidence bounds and represent the <u>joint</u> distribution of parameters.
  - _runname_MCMC_points.tab: A diagnostic file containing more information on sample points, including those rejected
  - _runname_MCMC_stats.tab: A diagnostic file containing convergence metrics

Using the Sample for Sensitivity Runs

- **Plug the _MCMC_sample.tab file in as a Sensitivity simulation**



Using the Sample for Sensitivity Runs



Bayesian System Dynamics

- **Bayes Rule: P(A|B) = P(B|A)*P(A)/P(B)**

Posterior

P(Params|Data)

= P(Data|Params) * P(Params) / P(Data)

Likelihood Prior Ignore

Implementation: combine calibration optimization or MCMC with priors that capture the state of knowledge about parameters.

Priors

- **No priors = uniform priors**
  - This is essentially what we've been doing so far
  - It's not always a good choice, *but* if you have lots of data, it probably doesn't matter.
- **Non-informative or Maximum Entropy priors**
  - Contribute as little information as possible, i.e. assume maximum ignorance a priori
  - For a scale parameter like a time constant, this is -LN(param) for positive parameters
- **Informative priors**
  - If you – or experts or literature – have some opinion about a parameter, you can use a subjective probability distribution to characterize that

Example

- **Suppose we think from other information that wolves live for about 7 years**

The life spans of wild wolves vary dramatically. Although the average lifespan is **between 6 and 8 years**, many will die sooner, and some can reach 13. Wolves in captivity can live up to 17 years. Apr 13, 2012

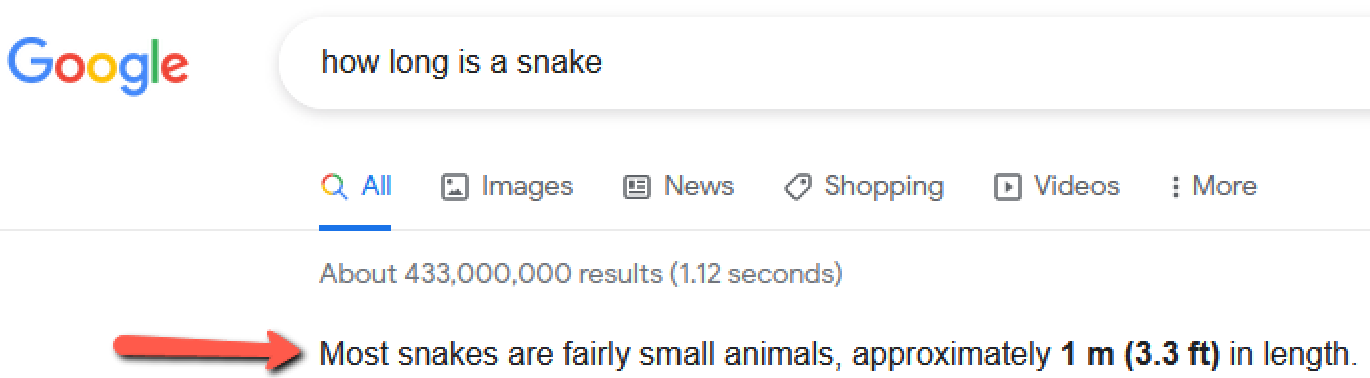
https://www.pbs.org › wnet › river-of-no-return-gray-wol...

River of No Return | Gray Wolf Facts | Nature - PBS

- **We could capture this in the model with a prior on the wolf mortality rate**

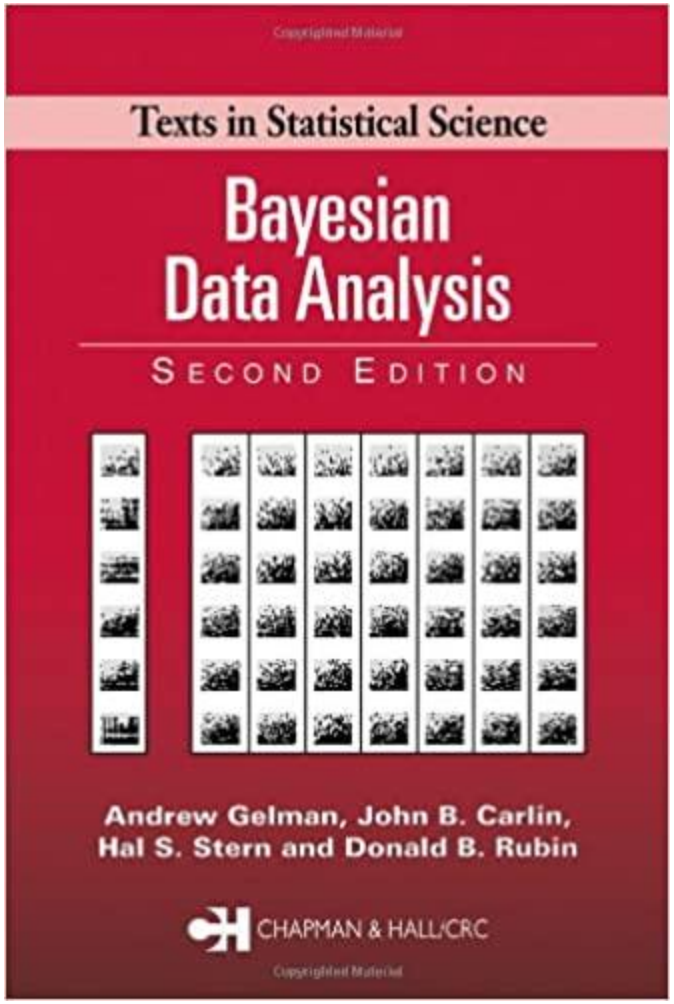
Likelihood for Priors

- **If our belief is Normal (Gaussian):**
- **Likelihood =** $\frac{1}{\sigma\sqrt{2\pi}}\, e^{\boxed{-\left(\frac{param-prior}{\sigma}\right)^2 \big/ 2}}$
- **For an MCMC log likelihood, we only need the last term**
- **σ represents our belief about the plausible variation in the prior**



Other Choices

- **Noninformative scale parameter**
  - -LN(parameter)
- **Interval variables**
  - Noninformative: Haldane or Jeffreys
  - Informative: Beta
- **Subjective**
  - Draw something in a lookup



Lifespan Prior

- **In the first order model, Lifespan = 1/Wolf Mortality Rate**
  - In the data generator, mortality rate = .08/year = 12.5 year lifespan, so there will be some conflict between our prior and the "truth"
- **We could use the Normal (Gaussian) distribution to express our prior, something like:**
  - Wolf Mortality Prior
    = -1/2*{(1/Wolf Mortality Rate – Wolf Lifespan Belief)
    / Wolf Mortality Confidence}^2
  - "Wolf Mortality Confidence" is the standard deviation, in years, expressing our belief about how widely lifespan might vary
- **Normality probably isn't the optimal choice, because it admits negative values; instead use Lognormal:**
  - Wolf Mortality Prior
    = -1/2*{LN(Wolf Mortality Rate*Wolf Lifespan Belief)
    / Wolf Mortality Confidence}^2
  - "Wolf Mortality Confidence" is the standard deviation of our belief, expressed as a fraction of the central value

My Typical Playbook

- Build/refine structure
- Load data
- Create an interface view with model-data comparisons
- Do some hand calibration to see what parameters are interesting
- Do a quick & dirty calibration
  - Weight payoff with log transform and wild guesses at fractional errors
- Evaluate fit, work with model more, ponder what is really problematic or uncertain
- Develop a more carefully weighted payoff, consider Kalman filtering, priors
- Do MCMC to generate a confidence sample
- Do sensitivity runs based on the sample
- Design policies
- Test policies deterministically
- Do policy experiments with sensitivity

VENTANA