



Vensim[®] Software

Linking systems thinking to powerful dynamic models

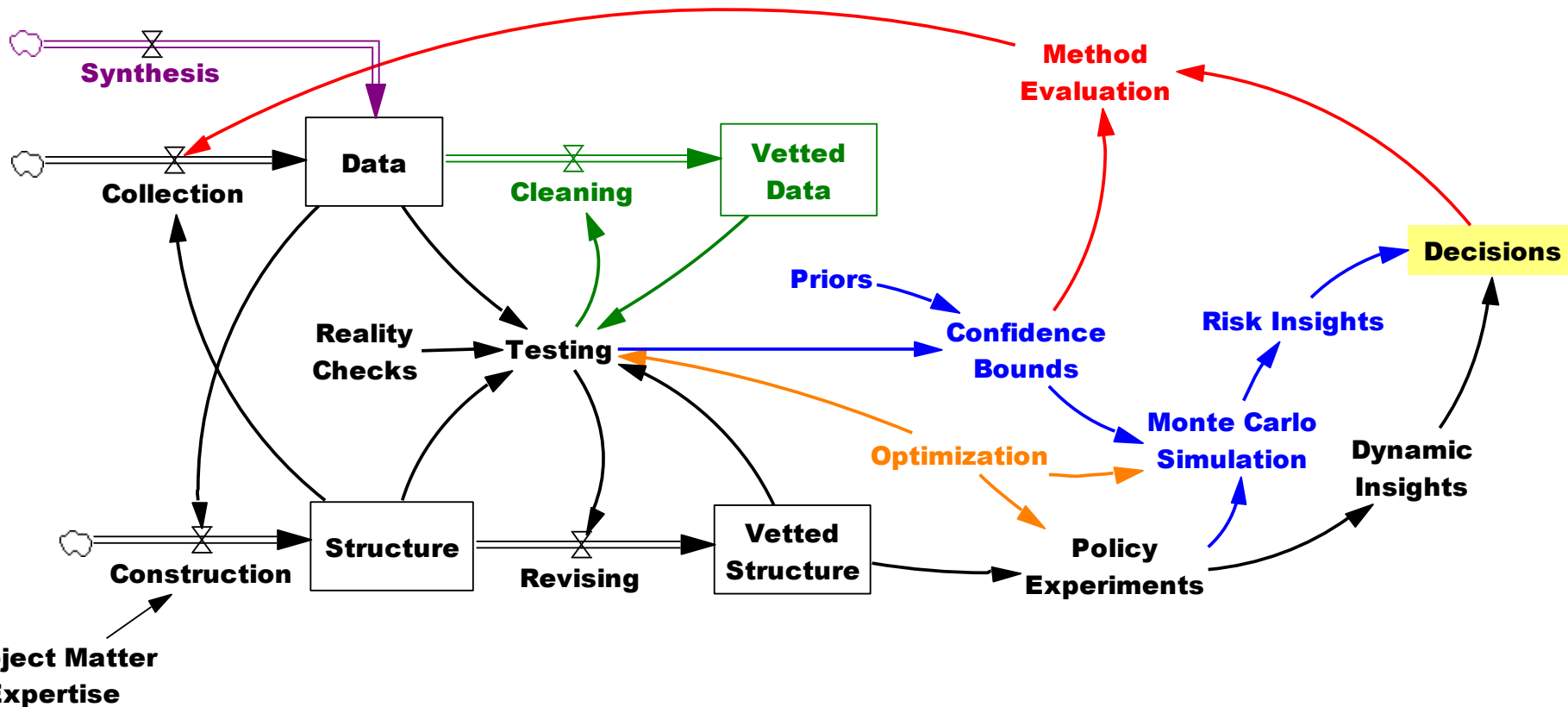
Optimization with Vensim

Tom Fiddaman
2016

Stylized Sequence

- **Build model & collect data**
 - **Run the usual structural tests**
 - **Calibrate**
 - **Run policy experiments on the best-guess model**
 - **Develop distributions for uncertain inputs from calibration insight and a priori information**
 - **Look for robust policies under uncertainty**
 - **Determine what data would resolve uncertainty**
 - **Determine what data would be most valuable**
- Analysis often stops here
- **Iterate!**
 - **If possible, prototype the whole process with synthetic data**

The Big Picture



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 (need compiled simulation, overnight runs)**

Orientation

- Open **BALCS for Analysis A.mdl** and make a test run
- **Sterman/Henderson/Beinhocker/Newman model*** of competitive dynamics with learning curves
 - Conventional wisdom: grow as fast as possible to outrun competitors down the learning curve
 - Dynamic insight: boom-bust markets and capacity acquisition lags make that advice dangerous
- **Take a few minutes to familiarize yourself with the model**
 - Play with the *Industry Controls* view – how do parameters affect the behavior?
 - Play with the *Strategy Controls* and *Financial Controls* views – how much money can you make?

* J. D. Sterman, R. Henderson, E. D. Beinhocker and L. I. Newman (2007) Getting Big Too Fast: Strategic Dynamics with Increasing Returns and Bounded Rationality. *Management Science*, 53, 683-696.
http://jsterman.scripts.mit.edu/On-Line_Publications.html#2007Getting

Strategy

Payoffs		Them	
		Conservative	Aggressive
Us	Conservative	Slow Market	They Win
	Aggressive	We Win	Boom & Bust

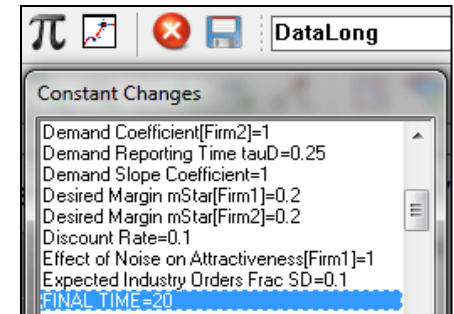
Part 1

Generating Synthetic Data

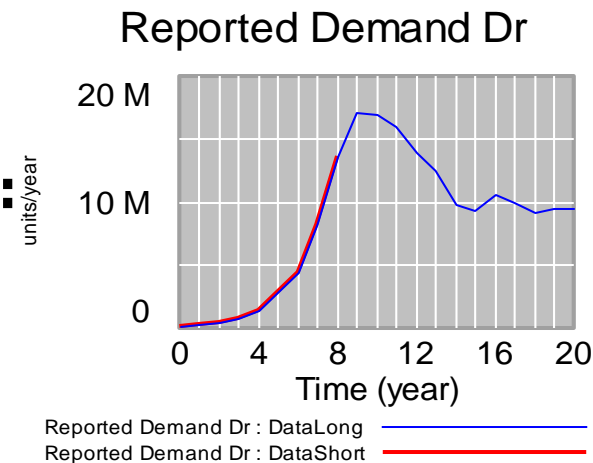
Generating Synthetic Data

- Set your run name to **DataLong**
- Start Synthesim
- Use the “Load Changes” button to open **DataGenerate.cin**
- Set **FINAL TIME** to 20 years (use the π button)

DataLong

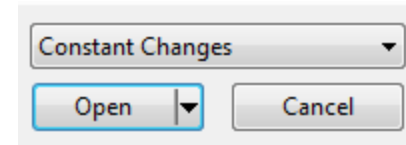


- Change the run name to **DataShort**
- Set **FINAL TIME** to 8 years
- You should have two runs that look like this: (*Calibration view*)



Side Trip: the Changes File

- Use **File>Edit File...** to open a .cin file in the text editor



- **It will look something like this:**

Adoption Noise SD = .1

Attractiveness Noise SD = .1

SAVEPER = 1

Order Reporting Noise SD = .05

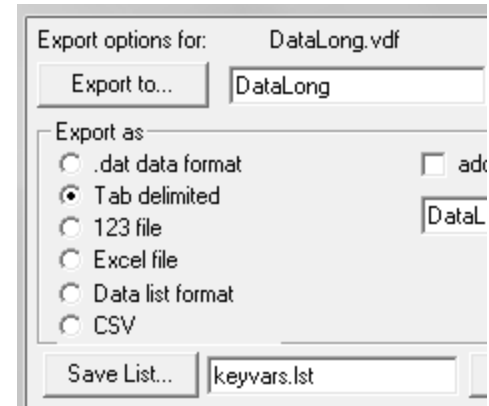
Demand Slope Coefficient = 1.35

Fractional Discard Rate delta = 0.1375

Propensity to Adopt from WOM beta = 1.4

- **These can be manually created in a text editor, or saved from the Changes tab of the Simulation Setup dialog or during Synthesim**

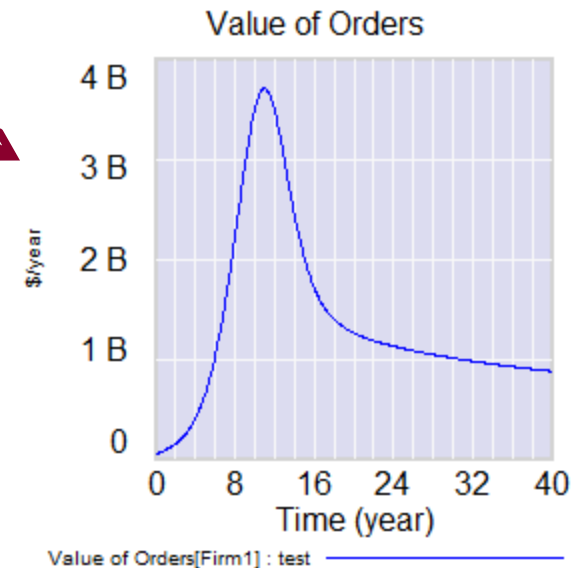
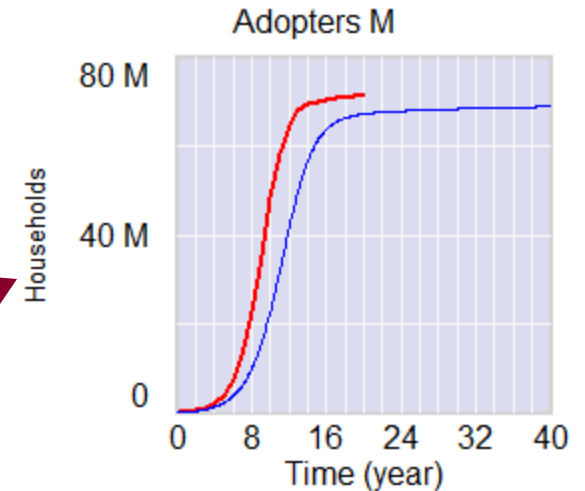
Exporting the Data



- **Select Model>Export Dataset... and choose DataLong.vdf**
 - Set options to **Tab delimited**, using Save List **keyvars.lst**
- **Use the text editor or a spreadsheet to inspect the resulting DataLong.tab**
- **Select Model>Import Dataset... and choose DataLong.tab**
 - Use the default settings.
 - Overwrite DataLong.vdf
- **What we've done is to restrict the set of variables available in the dataset to the few specified in keyvars.lst**
- **Optional: repeat for DataShort; try the .dat format**

Viewing the Synthetic Data

- Use the Datasets tab of the Control Panel to load DataLong and a normal run from the model
- Use the graph or table tools to look at a variable like Adopters that is in keyvars.lst
- Look at a variable like Value of Orders that is not in keyvars.lst



Part 1a

Driving Data (side trip)

Making Price Exogenous

- Save a copy of the model under a new name, like **BALCS with driving data.mdl**
- **Move to the Price view**
- **Edit the equation for Price P**
- **Change its type to "Data"**

Edit: Price P

Variable Information

Name Price P

Type Data Sub-Type Normal

Units \$/unit Check Units ☐ Supp

Group .BALCS with driving data E Min Max


Equations

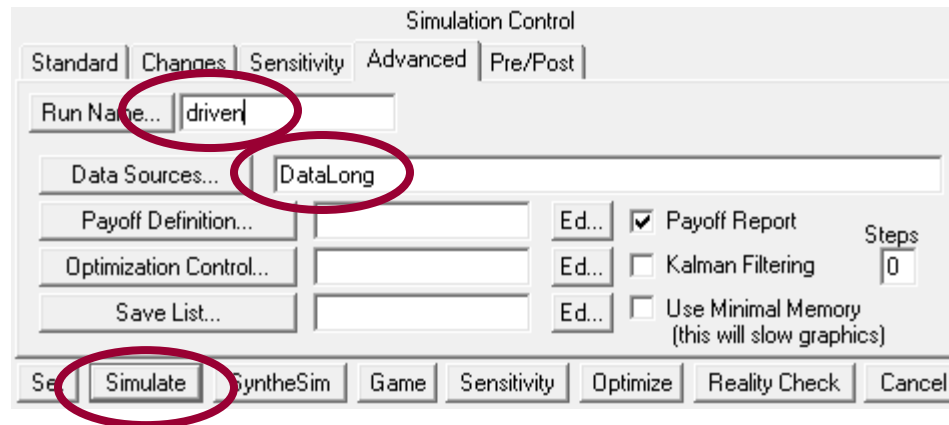
Subscript [Firm

☐ Except

- Notice that causal arrows (in this case, a flow pipe) influencing Price P disappear

Running with Driving Data

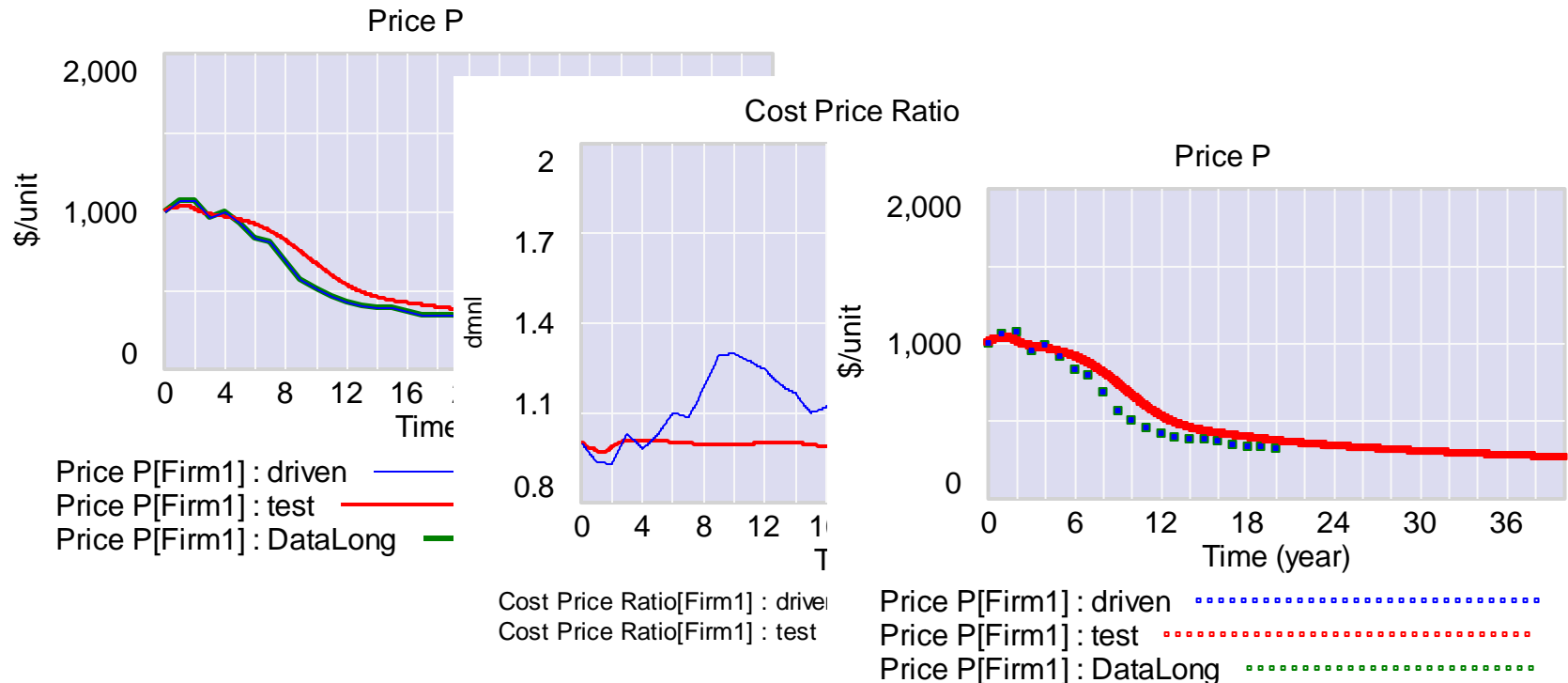
- Use **Model>Simulate** or the Sim Control button  to launch the simulation control dialog
- Switch to the **Advanced** tab and specify **DataLong** or **DataShort** as the input Data Source



- Choose a new run name.
- Simulate.

Inspecting the Results

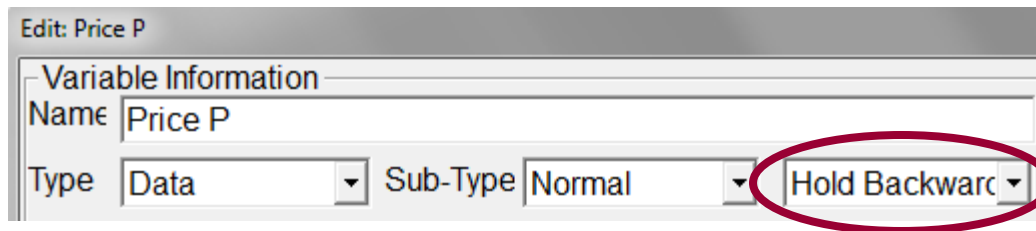
- **Compare Price P and downstream variables like Cost Price Ratio, with and without the driving data**



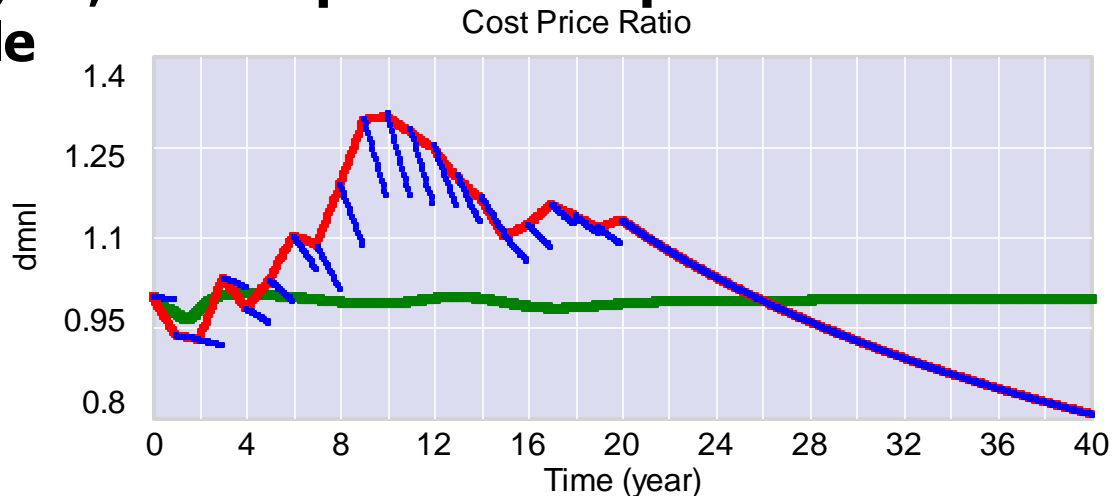
- **Right-click the Graph tool and set it to show “Dots Only” to see individual data points**

Interpolation Patterns

- Change the interpolation pattern for Price P to “Hold Backward” or “Look Forward”



- Run again, and inspect the output for a downstream variable



Cost Price Ratio[Firm1] : driven HOLD
Cost Price Ratio[Firm1] : driven
Cost Price Ratio[Firm1] : test

Driving Data from a Spreadsheet

- Launch the **DiscountRate.xlsx** spreadsheet
- Switch to the **Financials** view
- Edit the **Discount Rate** equation
- Change its type to **Data**, with sub-type **Equation**
- Use **GET XLS DATA** to link to the Disc Rate row in the data

	A	B	C	D
1	Year	2000		
2	Quarter	1	2	3
3	Month	0	3	6
4	Cost of Equity	0.03	0.05	0.07
5	Cost of Debt	0.05	0.05	0.07
6	WACC	0.04	0.05	0.07
7	Risk Premium	0.05		
8	Disc Rate	0.09	0.10	0.12

Edit: Discount Rate

Variable Information

Name

Type Sub-Type

Units ☐ Supplementary

Group Min Max

Equations

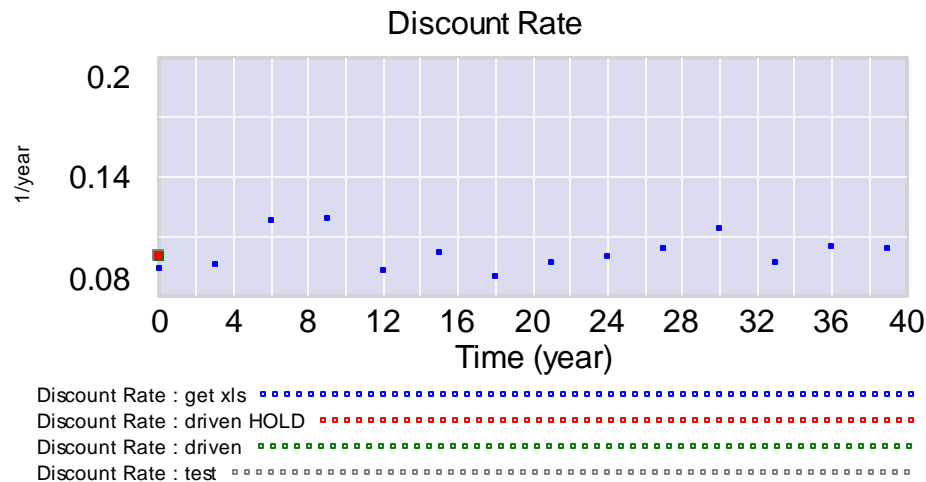
Subscripts

:=

- No spreadsheet? Use **GET DIRECT DATA** instead

Running with XLS data

- Spreadsheet must be open for GET XLS (or Vensim will launch it)
- File must be closed for GET DIRECT (Excel file locks prevent reading)
- Simulate the model & inspect the output



Data Strategies

1. Spreadsheet + GET

- Create one or a few multi-tabbed spreadsheets containing all the data (and constant inputs if desired)
- Reference with GET XLS or GET DIRECT
- Data are always “live”
- Links can be fragile (though use of named ranges helps)

2. Spreadsheet + file import to .vdf

- Create multi-tabbed spreadsheets, as above
- Create a master tab that aggregates all of the data in a format that’s easy to handle with File>Import...
- Automate the import with a command script on the Pre/Post tab of the simulation control

3. Data model

- Create a separate model to process data from various sources
- Use the output .vdf from the data model as input to the dynamic model

4. Database repository + ODBC import/export

Part 2 Calibration

- Close the driving data experiment and return to **BALCS for Analysis A.mdl**

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
- After convergence, evaluate the fit
 - Decide if the model can be rejected

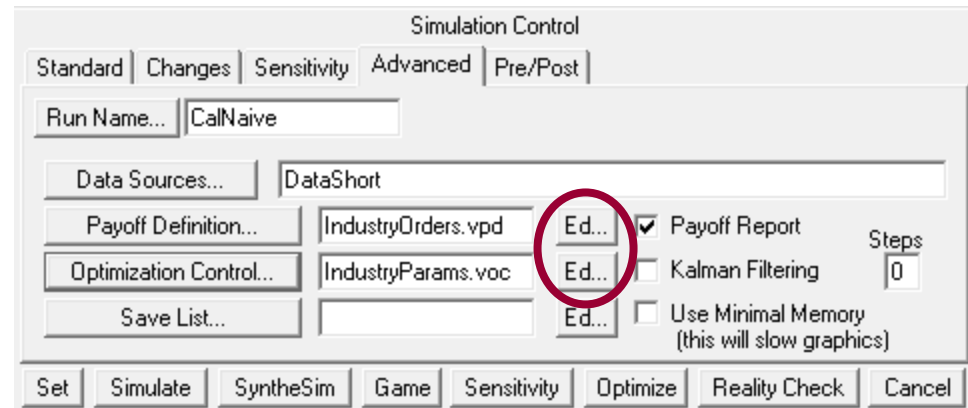
Naïve Calibration

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

Naïve Calibration

- Forget that we know the true market parameters, and see if we can estimate them from the data
- Use **Model>Simulate...** to open the Simulation Control
- Go to the Advanced tab
 - Choose an informative Run Name, like **CalNaiveShort**
 - Set the Data Source to **DataShort.vdf**
 - Set the Payoff Definition to **IndustryOrders.vpd**
 - Set the Optimization Control to **IndustryParams.voc**
- Hit the **Optimize** button

You can peek in the files using the Ed... buttons



Payoff (.vpd)


Payoff Definition

Payoff Definition. Edit the filename to save changes to a different control file

Filename:

Payoff Elements

Calibration: Normal: Always: None: Users\data facebook users/users weight
Policy: Normal: Initial: None: prior likelihood/1



Payoff Element

Payoff type

☒ Calibration ☐ Policy

Payoff details

Variable

Compare to

Weight (1/StdDev)

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

Distribution

Timing

Calibration Payoff Types

Payoff Element

Payoff type

☒ Calibration ☐ Policy

Payoff details

Variable: Users Sel

Compare to: data facebook users Sel

Weight (1/StdDev): users weight 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

Model Variable

Data Variable (optional if data names match)

Scale or Weight (interpretation depends on distribution)

Log transform?

Error distribution assumption & format

Error Distribution Assumptions

- **N – Normal**
 - Payoff is the sum of $(\text{model-data})^2 \times \text{weight}$
 - Weight typically = $1/(\text{standard error of measurement})$
 - Proportional to $2 \times \log \text{likelihood}$, as long as you're not estimating the weight
- **G – Gaussian**
 - Sum of $(\text{model-data})^2 \times \text{StdDev}/2 + \text{LN}(\text{StdDev})$
 - 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 $\text{ABS}(\text{model-data}) * \text{StdErr} / 2 + \text{LN}(\text{StdErr})$
 - StdErr 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 StdErr
 - Not as efficient as Gaussian, but resistant to contaminated data
- **Y – Cauchy**
 - Not really practical, unless noise is absurdly extreme
- **For most purposes, use Normal, Gaussian or Robust**

Policy Payoff Types

The screenshot shows a 'Payoff Element' dialog box with the following fields and annotations:

- Payoff type:** Radio buttons for 'Calibration' and 'Policy' (selected).
- Payoff details:**
 - Variable:** Text box containing 'prior likelihood'. Annotation: **Model Variable**.
 - Compare to:** Empty text box with a 'Sel' button.
 - Weight (1/StdDev):** Text box containing '1'. Annotation: **Weight (importance; sometimes thought of as a price)**.
 - Transform:** Dropdown menu set to 'None'. Annotation: **Log transform?**.
 - Distribution:** Dropdown menu set to 'Normal'.
 - Timing:** Dropdown menu set to 'Initial'. Annotation: **Timing – Always compute, or Initial or Final only**.
- Buttons:** 'OK' and 'Cancel' at the bottom.

Below the dialog box, there is a red text annotation: **Weight (importance; sometimes thought of as a price)**.

The Payoff File as text

***C**

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

Reported Demand Dr/1

Model variable/weight

If a calibration comparison is to a different variable, this may look like:

Reported Demand Dr|Demand Data/1

The weight can also be a variable.

Subscript ranges are OK, as long as they match.

Optimization Control File (.voc)

Method & Settings

Parameters & Bounds

Optimization Control

Filename
Optimization Control. Edit the filename to save changes to a different control file
Filename:

Optimizer

Optimizer	<input type="text" value="MCMC"/>	Stochastic	<input type="text" value="No"/>	Seed	<input type="text" value=""/>
Limit	<input type="text" value="80000"/>	Jump	<input type="text" value="0.05"/>	Floor to Temp	<input type="text" value="1"/>
Burnin	<input type="text" value="70000"/>	Delta	<input type="text" value="0.0001"/>	Temp	<input type="text" value="2"/>
# Chains	<input type="text" value="2"/>	# Pairs	<input type="text" value="2"/>	Schedule	<input type="text" value="None"/>
Outlier	<input type="text" value="0.05"/>	X Over	<input type="text" value="0.2"/>	Cooling	<input type="text" value="1000"/>
Init method	<input type="text" value="Default"/>	Payoff Type	<input type="text" value="Log likelihc"/>	Epsilon	<input type="text" value="0.01"/>
Gamma	<input type="text" value="1"/>	Record	<input type="text" value="All points"/>		

Choose optimization parameters

0.1<=init users<=10
0.1<=unsaturated growth rate r<=2
.25<=saturation exponent<=4
1<=carrying capacity K<=10000

<= = <=

Model value of constant -- =

The Optimization Control File as text

:OPTIMIZER=Powell

:SENSITIVITY=Off

:MULTIPLE_START=Off

<bla bla bla – algorithm control settings>

:VECTOR_POINTS=25

List of parameters to optimize:

.5<=Demand Slope Coefficient =1.35<=2

0<=Fractional Discard Rate delta =0.1375<=1

0<=Propensity to Adopt from WOM beta =1.4<=4

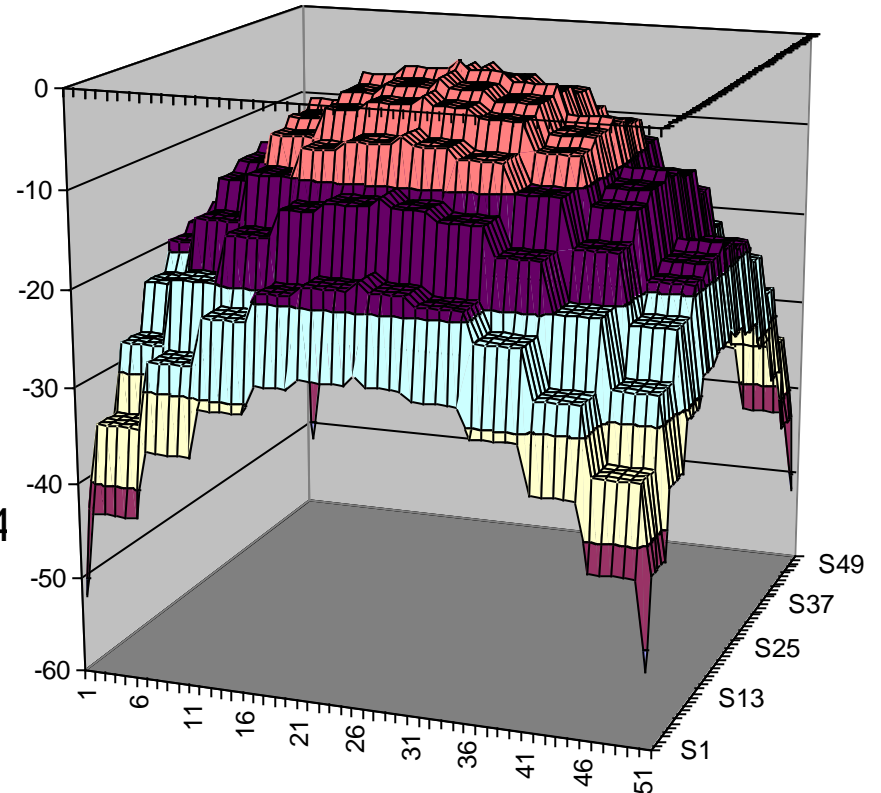
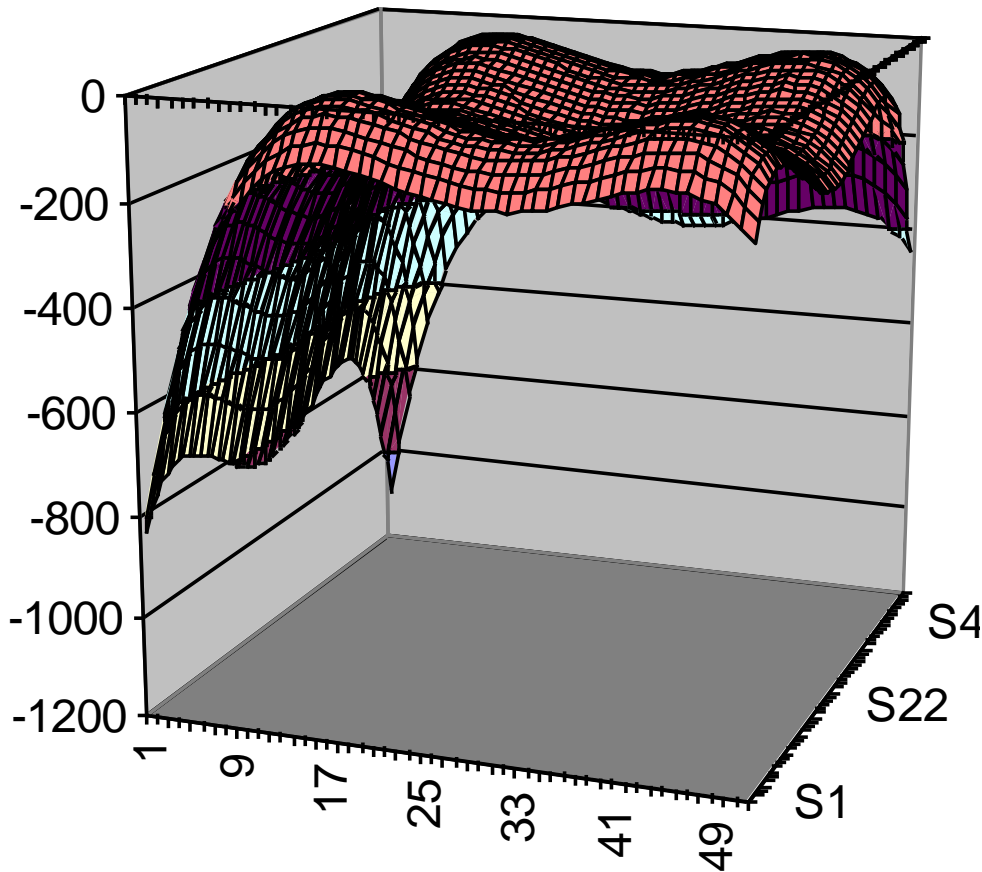
Min <= Variable Name = Initial Guess <= Max

Subscript ranges are OK. Initial Guess is often omitted.

What are we doing?

- **The Data Source locates the input data to be compared to model variables (this could also be a model variable from GET XLS or ODBC sources)**
- **The Payoff specifies what data series to match, and how to weight each one**
- **The Optimization Control file specifies 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**

Hill climbing is not always straightforward



Approaches to Difficult Payoff Surfaces

- **Try to restructure the model, parameter inputs, or payoff, to avoid relationships that cause ridges or other perverse features**
- **Failing that,**
 - Use Random Multistart to optimize from many starting points
 - Use random or grid search to explore the space without optimizing
 - Use vector searches or visualization tools to try to get some insight into the payoff surface

Maximum Likelihood

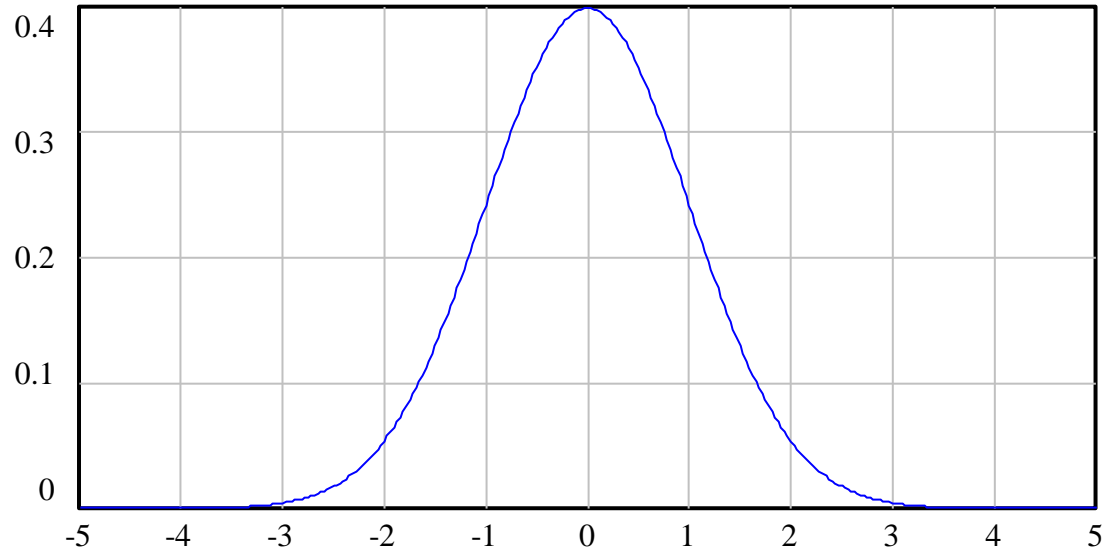
- **Choose the value of parameters that maximizes the likelihood of observing the data given the model**
- **This is called a Maximum Likelihood Estimator (MLE)**
- **Suppose there is more than one observation**
 - Then the likelihood is the product of the individual likelihoods for each data point
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

Likelihood Surface Normal (Gaussian) errors

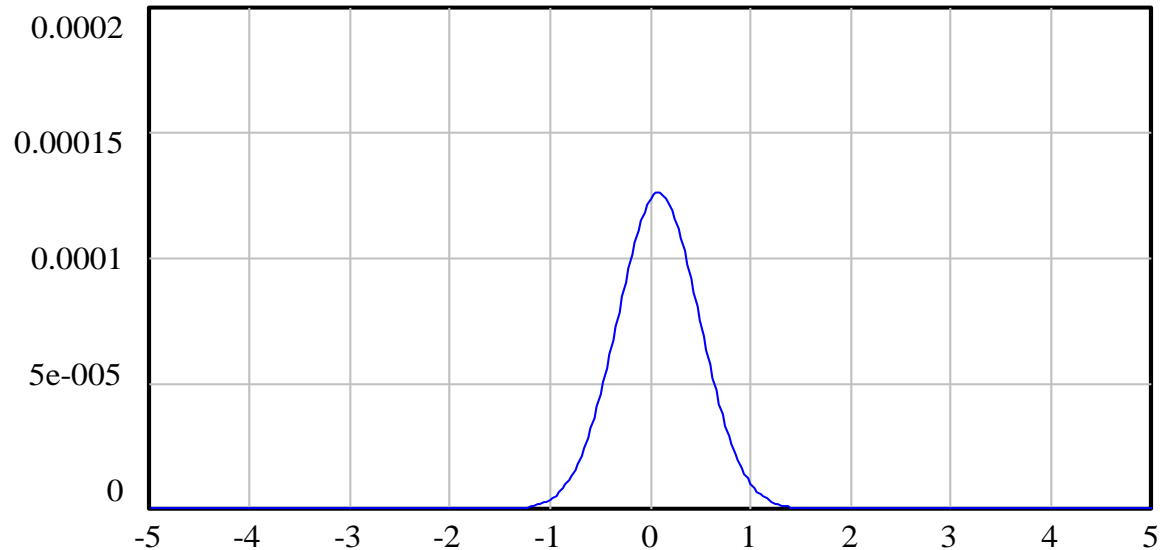
- **Likelihood** = $\frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(model - data)^2}{2\sigma^2}}$
- **σ represents the standard error associated with a data point, corresponding with the weight assigned in Vensim (or its inverse)**
- **Log(likelihood) simplifies to $-(model - data)^2$ – hence the motivation for ordinary least squares fitting (OLS)**

Adding data shrinks the likelihood peak

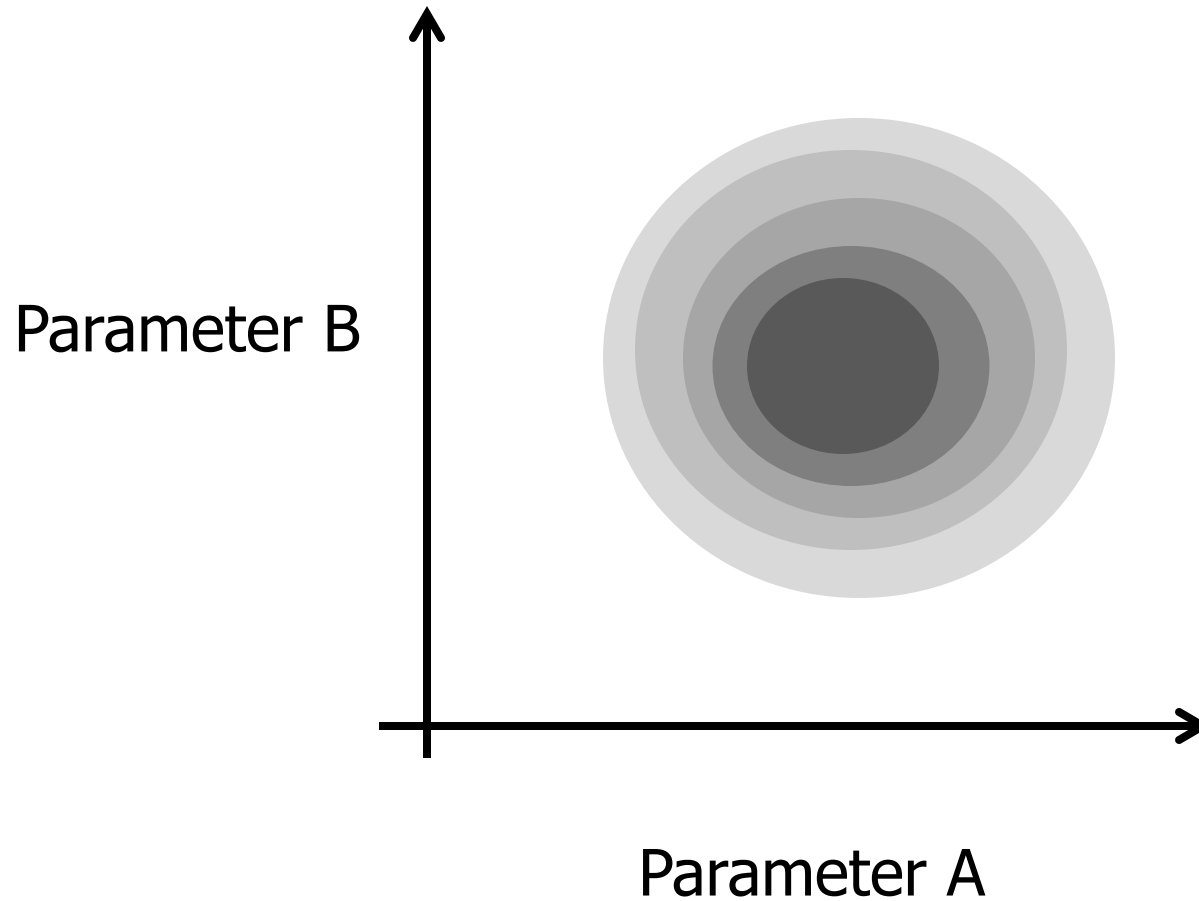
One Point



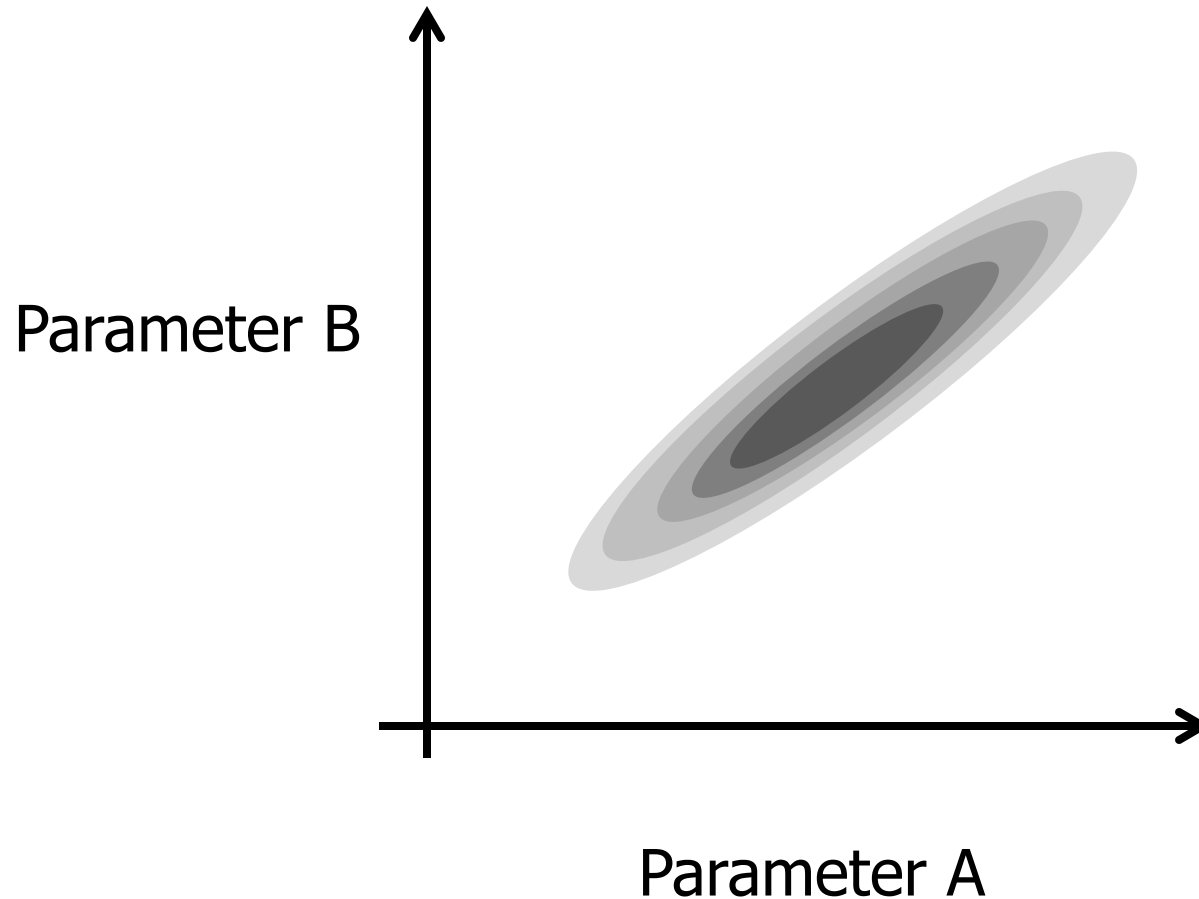
Several Points



Multidimensional Likelihood



Multidimensional Likelihood Related Parameters



Naïve Calibration With More Data

- Repeat the last calibration, but use **DataLong** as the Data Source (don't forget to pick a new run name)
- How do the parameters for the two estimates compare?
 - Use the **Runs Compare** tool, or
 - Use **File>Edit...** to examine the **.out** files created by the optimizer
- How do the market forecasts for the two estimates compare?
- What would happen if we only had the short data to work with for some real situation?

Refining the Calibration

- **Switch from normal to lognormal errors**
 - Assume that errors in reporting Industry Orders are roughly proportional to the scale of the data, rather than absolute
- **Weight the estimates with a wild guess about the scale of errors (10%)**
 - We could estimate this from model-data residuals
- **Add a prior on the Discard Rate**
 - Presumably we have some idea about the durability and context of use of the product
 - Posterior density (from Bayes' rule)
$$p(\text{params}|\text{data}) = p(\text{params}) * p(\text{data}|\text{params}) / p(\text{data})$$

Why weight the payoff?

- **To recognize varying quality of different data series**
- **For computation of confidence bounds**
 - A properly-weighted log-likelihood calibration payoff is distributed Chi-squared with one degree of freedom
 - The expected value is the number of data points
 - Varying the payoff by 3.84 yields a 95% confidence bound
- **Caveats:**
 - Autocorrelation
 - Structural uncertainty

How do you determine σ ?

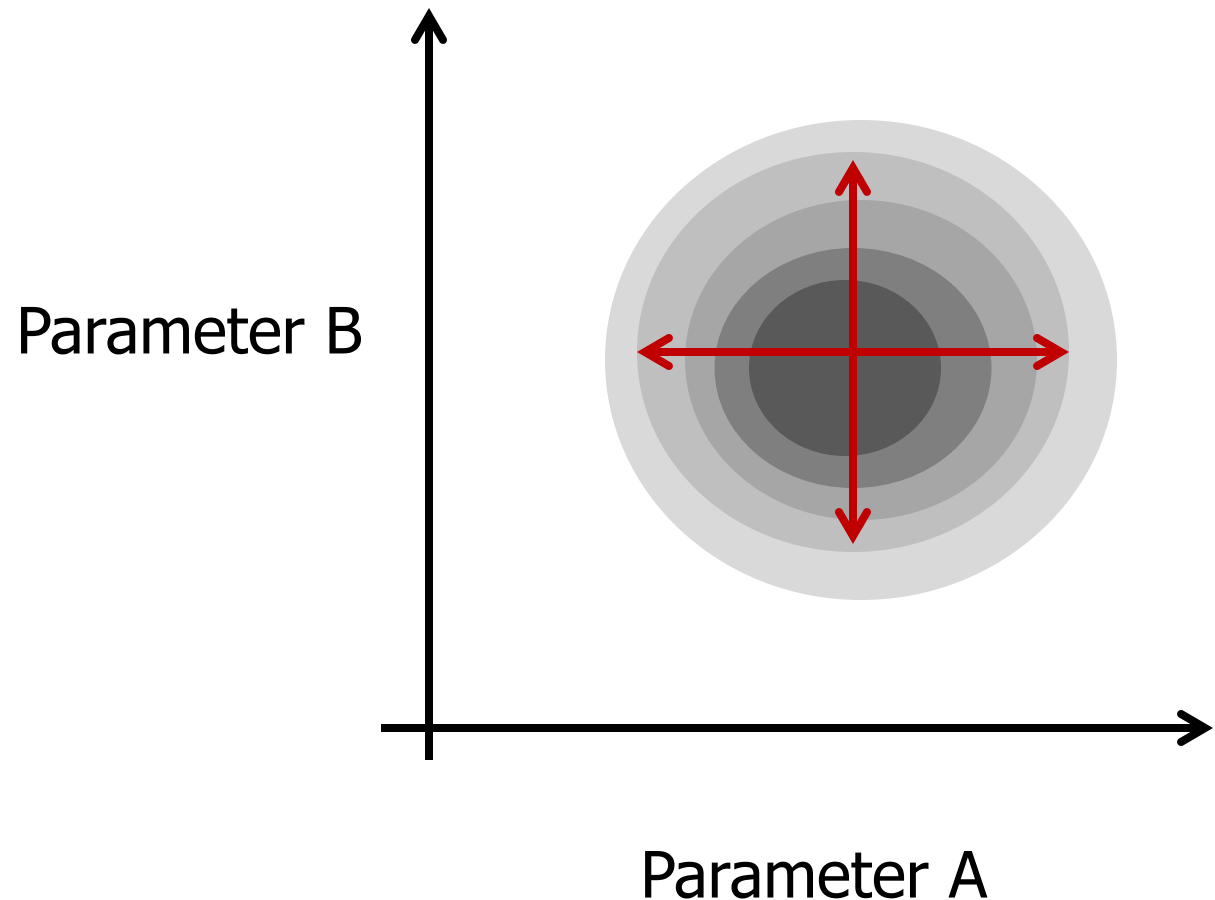
- **Guess:**
 - “plus or minus x%”
 - Standard deviation of the data (if stationary)
- **Iterate:**
 - Run the model
 - Look at the payoff or the residuals
 - Adjust the error toward what you observe
- **Estimate:**
 - Include the error or weight as an optimization control parameter
 - Requires extra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

Confidence Bounds

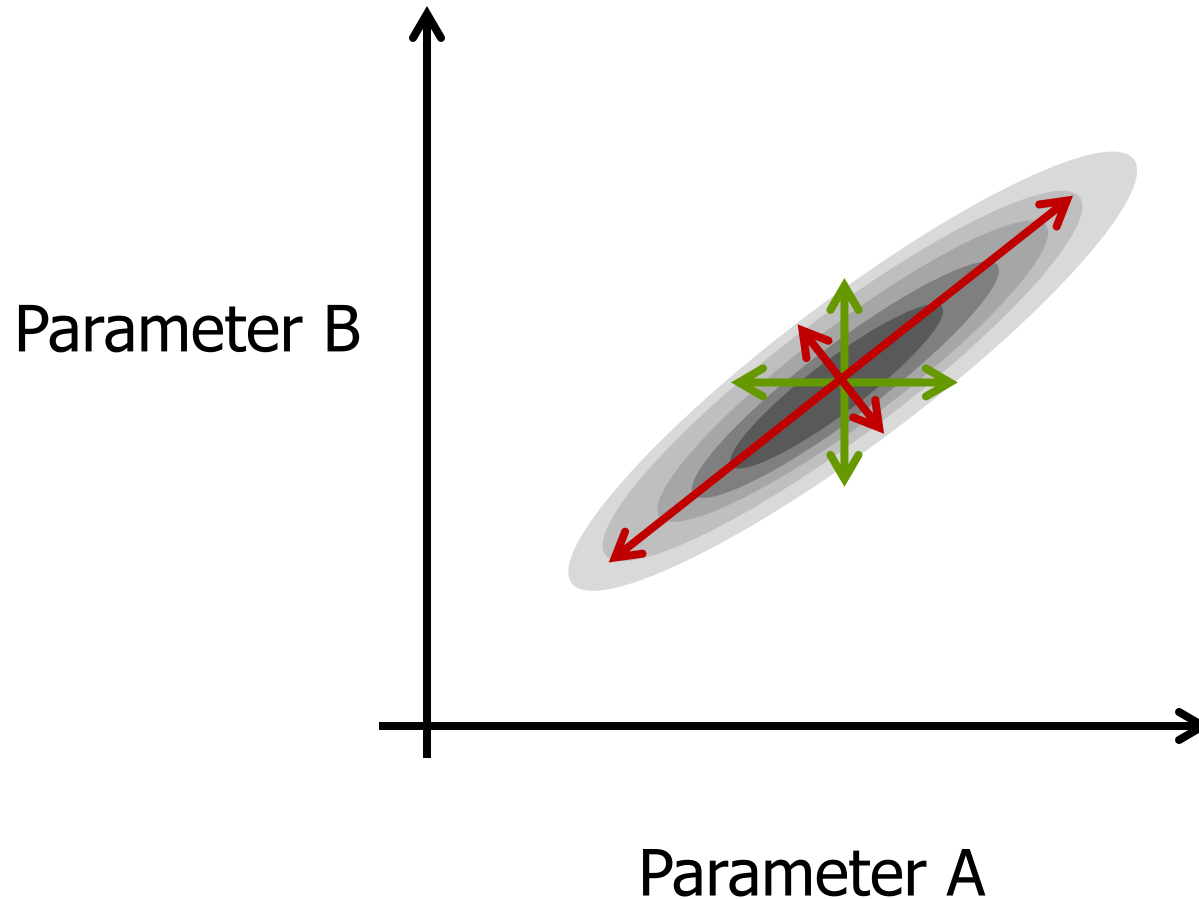
- **Normal (Gaussian) Likelihood** = $\frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(\text{model}-\text{data})^2}{2\sigma^2}}$
- **Log Likelihood** $\approx -\ln \sigma - \frac{(\text{model}-\text{data})^2}{2\sigma^2}$
- **Therefore a weighted log-likelihood calibration payoff is a sum of squares, distributed Chi-squared with one degree of freedom**
- **The expected value is the number of data points**
- **Varying the payoff by 3.84 (the ChiSq critical value at 95%) yields a 95% confidence bound**
- **Note: in some cases (depending on whether the /2 term is applied to the sum), you need to vary the payoff by $3.84/2 = 1.92$ (see the Help system for details)**

Standard Vensim payoff value sensitivity

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

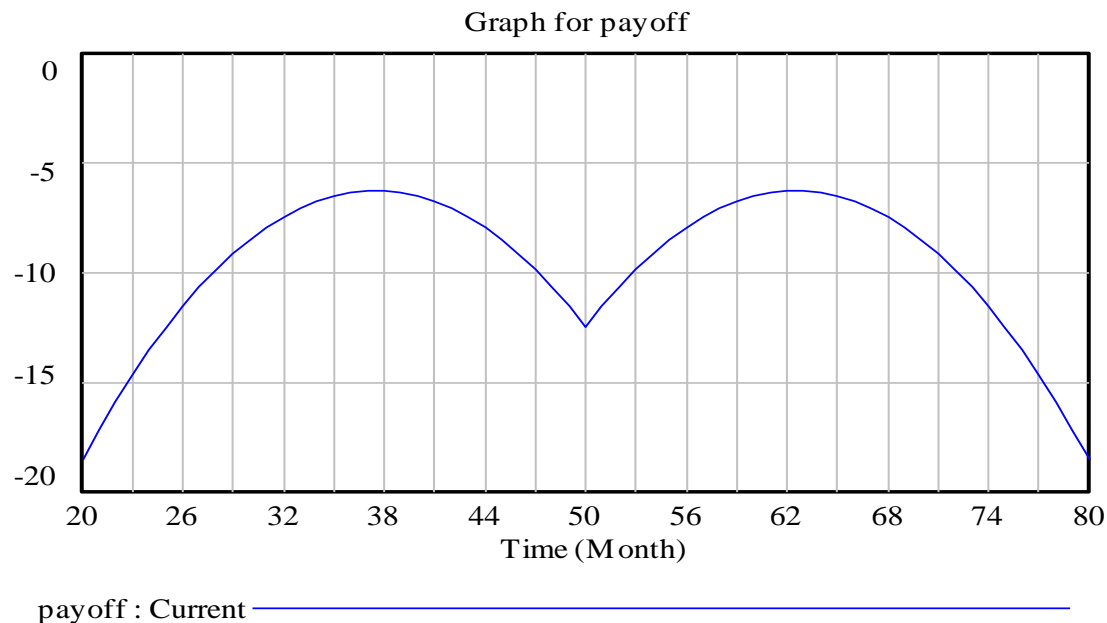


Misses off-axis ellipsoids!



Unimodality, Smoothness

- If not then the confidence bounds can be misleading

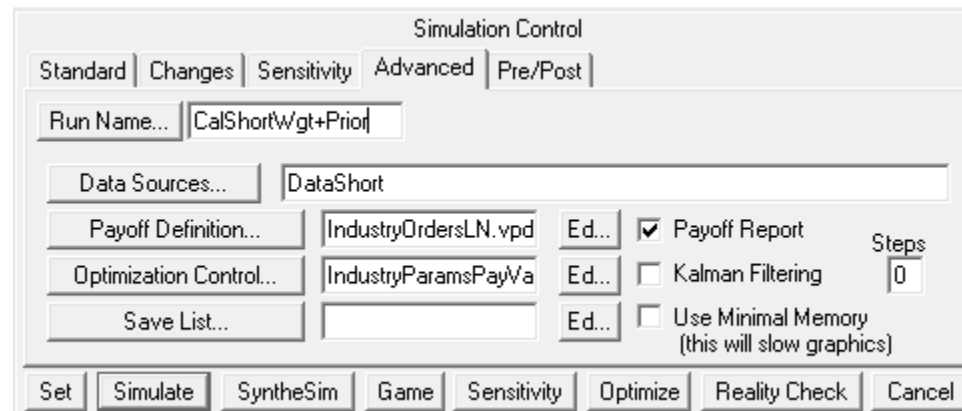


ReCalibration

- Use **Model>Simulate...** to open the **Simulation Control**
- **Go to the Advanced tab**
 - Choose an informative Run Name
 - Set the Data Source to DataShort.vdf
 - Set the Payoff Definition to IndustryOrdersLN.vpd
 - Set the Optimization Control to IndustryParamsPayVal.voc
- **Hit the Optimize button**
- **Repeat using DataLong**

Peek in the files, but **DON'T SAVE CHANGES** to
IndustryOrdersLN.vpd

(Use File>Edit... for modifications)



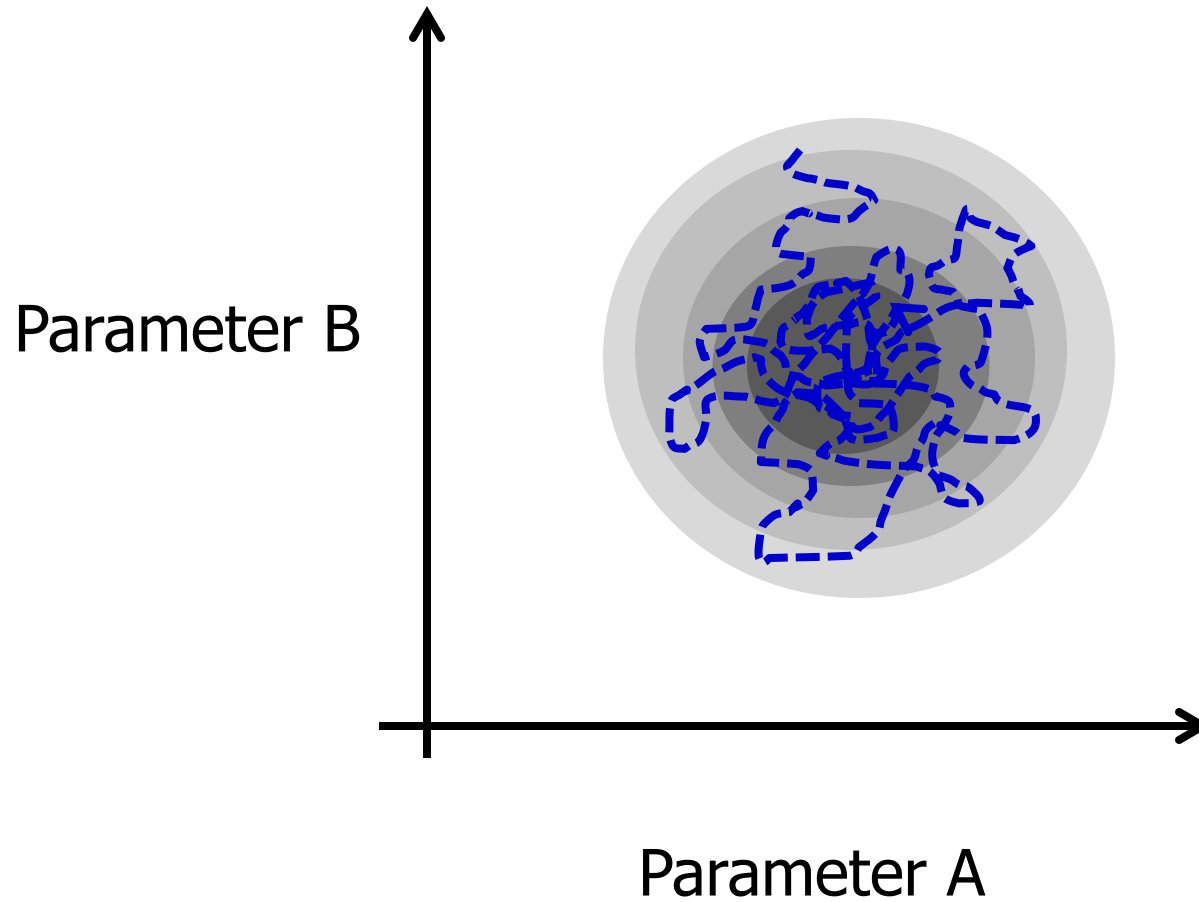
Results

- How do the forecasts of the refined calibrations compare?
- What are the estimates for the Discard Rate parameter?
 - Does the data or the prior dominate the result?
- Use File>Edit... to take a look at the **<run name>_sensitive.tab** files containing confidence bounds on parameters
- Use File>Edit... or Excel to inspect the payoff reports (***.rep files**)

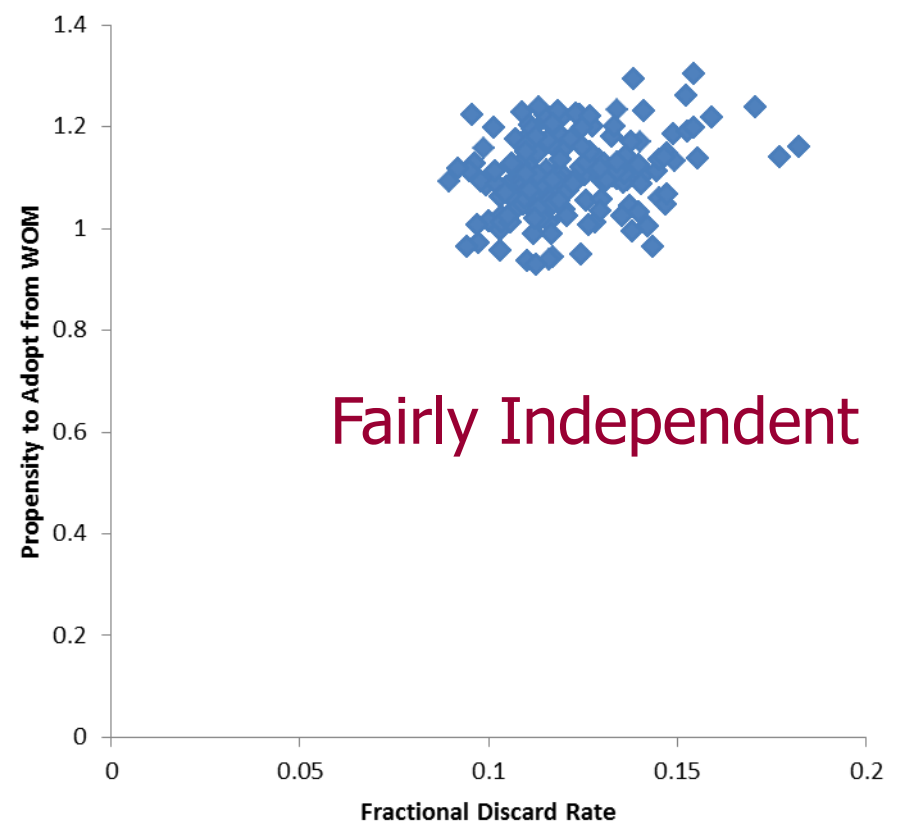
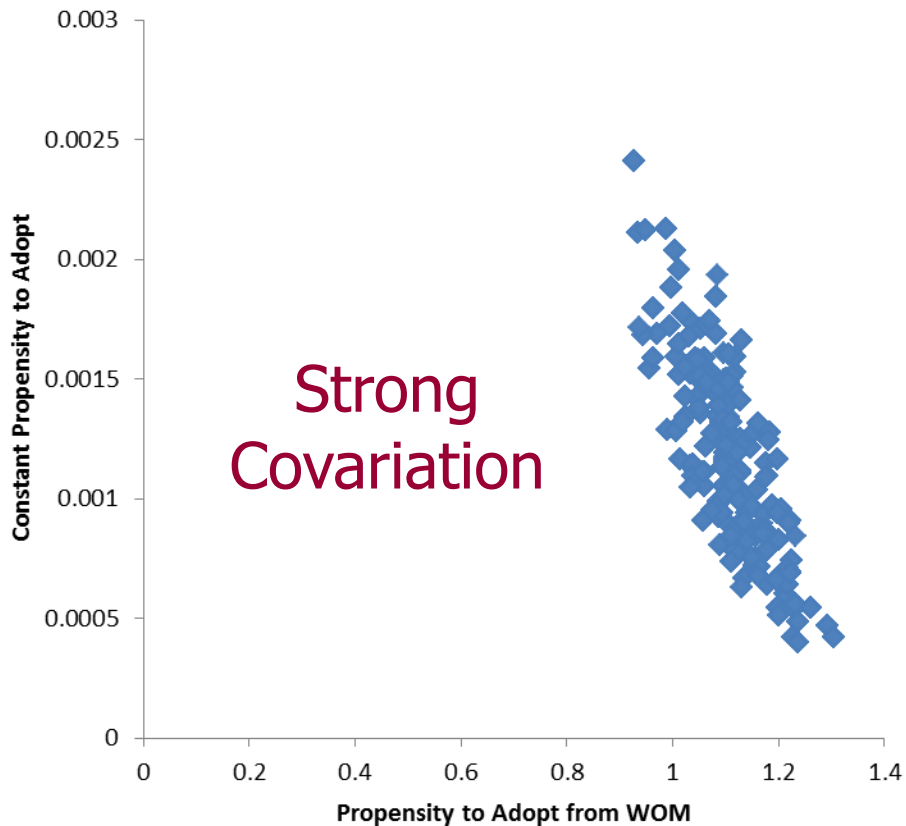
Alternate Approach to Confidence Bounds

- **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 Results

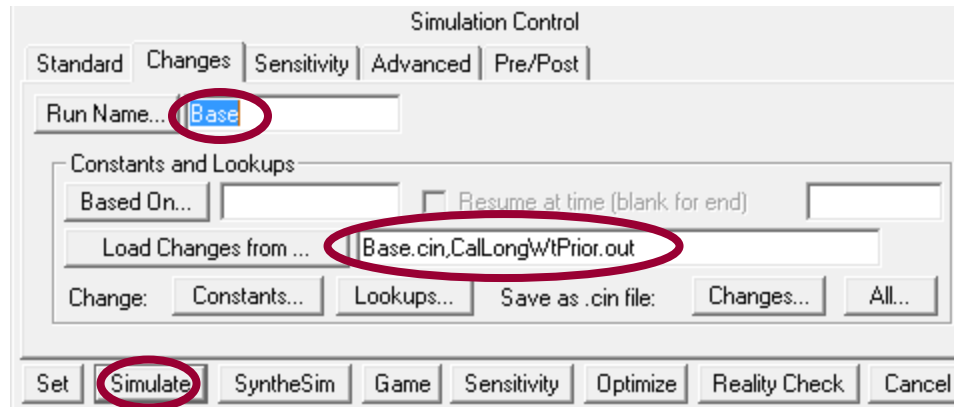


Part 3

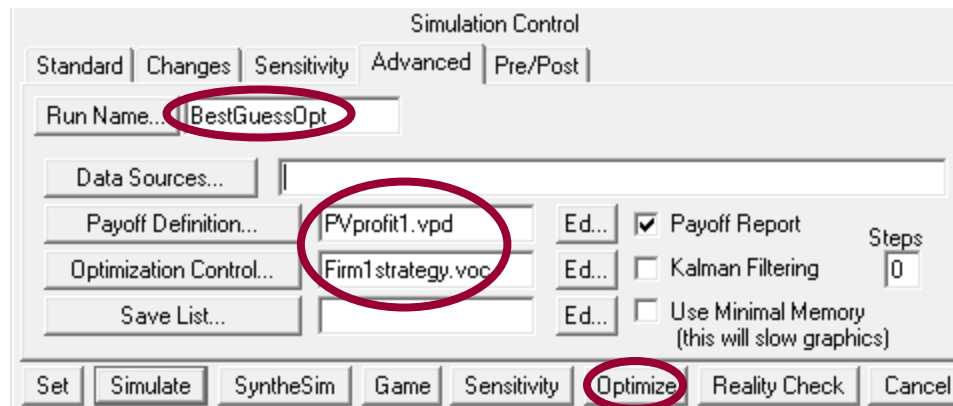
Policy Experiments with Uncertainty

Best-Guess Deterministic Policy

- Do a “base” run, using the best-guess calibrated market parameters by loading your .out file as a changes file:



- Now repeat, but optimizing to maximize profit by varying Firm1's strategy:



Key Questions

- **Does your deterministic optimal policy, developed using your best guess about the market, against a conservative competitor, work in general?**
- **Is there a policy that would work better on average?**

Payoffs accumulate, like stocks

- Notice that the payoff file (**PVprofit1.vpd**) uses the flow of **Discounted Profit[Firm1]**
- The payoff computation process will accumulate Discounted Profit over time, so that the total payoff is effectively a stock, just like **PV Profit[Firm1]** on the **Financials** view
- If you want to use the final value of a stock as your payoff, you can use the ***PF** keyword, but the usual ***P** flow payoff computation may have higher numerical precision

Notes on Policy Optimization

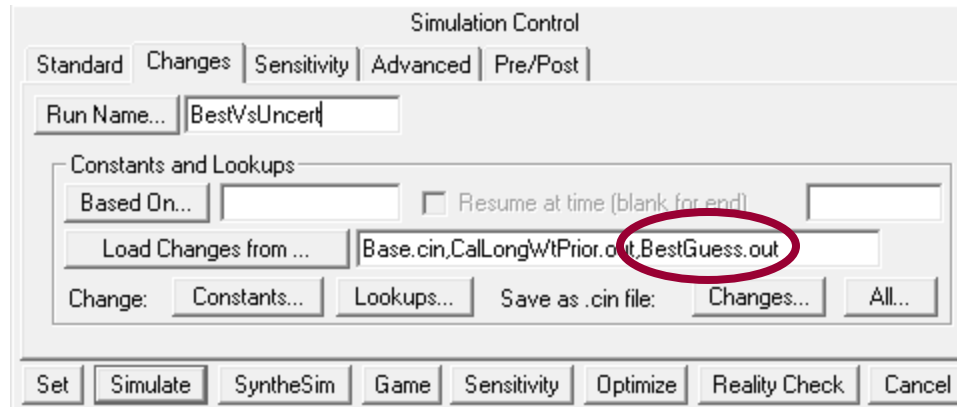
- **General Notion**
 - Treat the model like a function that transforms parameters to a single score
 - Figure out how to get the maximum score
- **The payoff is seldom obvious for real problems**
 - Conflicting goals (profit, employee satisfaction)
 - Questions about weight
 - Questions about timing (discounting)

Conflict Resolution

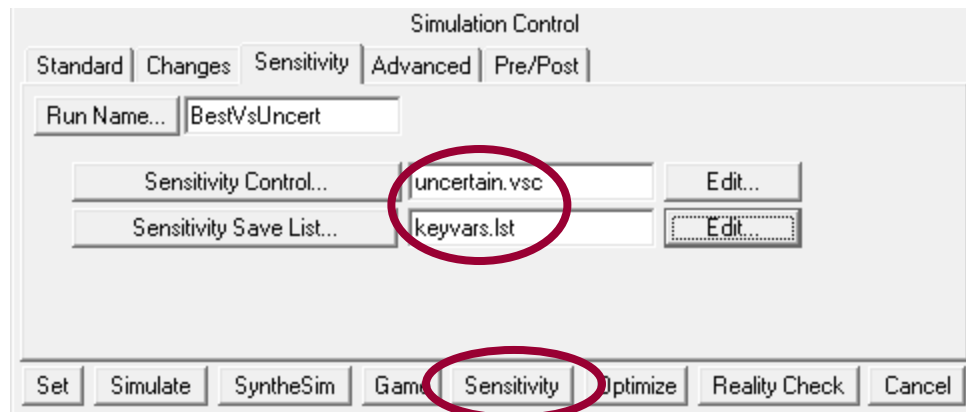
- **People are often concerned with indicators of future progress**
 - Balanced scorecard
 - Green GDP measures
 - Consumer confidence
- **A good model can be more direct**
 - Time value of performance
 - Simulate as long as you want
 - Refine goals based on outcomes

Test the Best Guess Against Uncertainty

- Add the .out file from your deterministic policy to the list of changes:



- Switch to the sensitivity tab, and add a control file and savelist, then run:

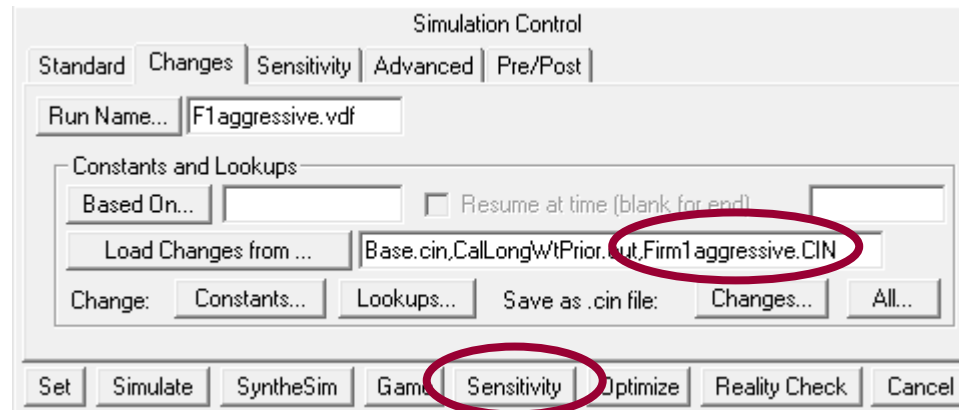


What are we doing here?

- **The sensitivity control file – `uncertain.vsc` – contains a list of parameter distributions**
 - Market adoption parameters: distributions (notionally) informed by the confidence bounds we estimated
 - Other distributions from priors about the world (subject matter expert consultation or wild guesses)
 - Other market parameters (consumer choice)
 - Noise parameters
 - Competitor strategy (drivers of pricing and capacity decisions)
- **The savelist, `keyvars.lst`, limits the list of variables to store (preventing huge files)**
- **Note that we might be understating our true uncertainty because we ignore:**
 - Covariance of non-orthogonal parameters
 - Structural uncertainty



Test an Aggressive Strategy Against Uncertainty

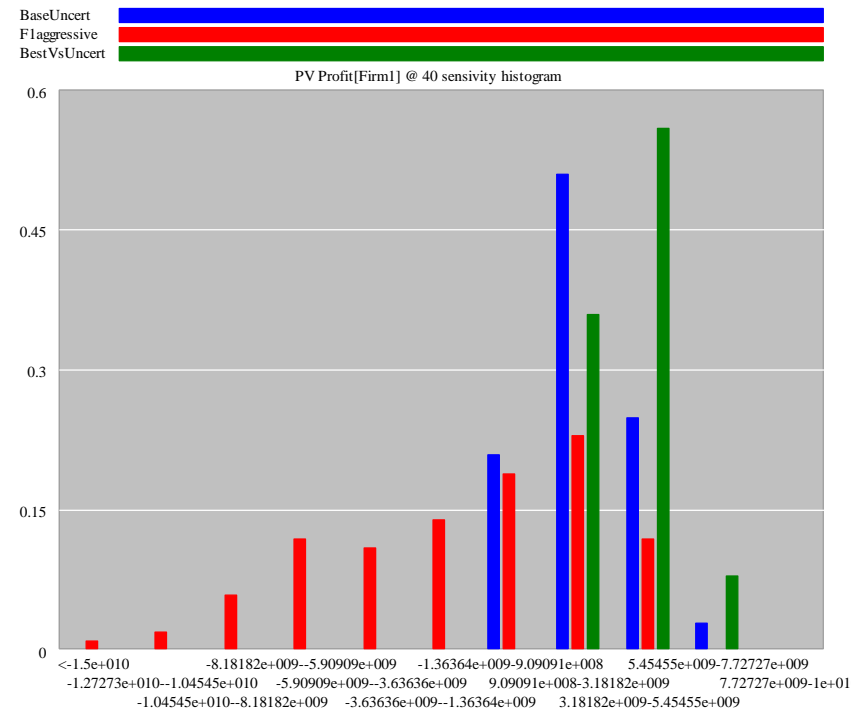
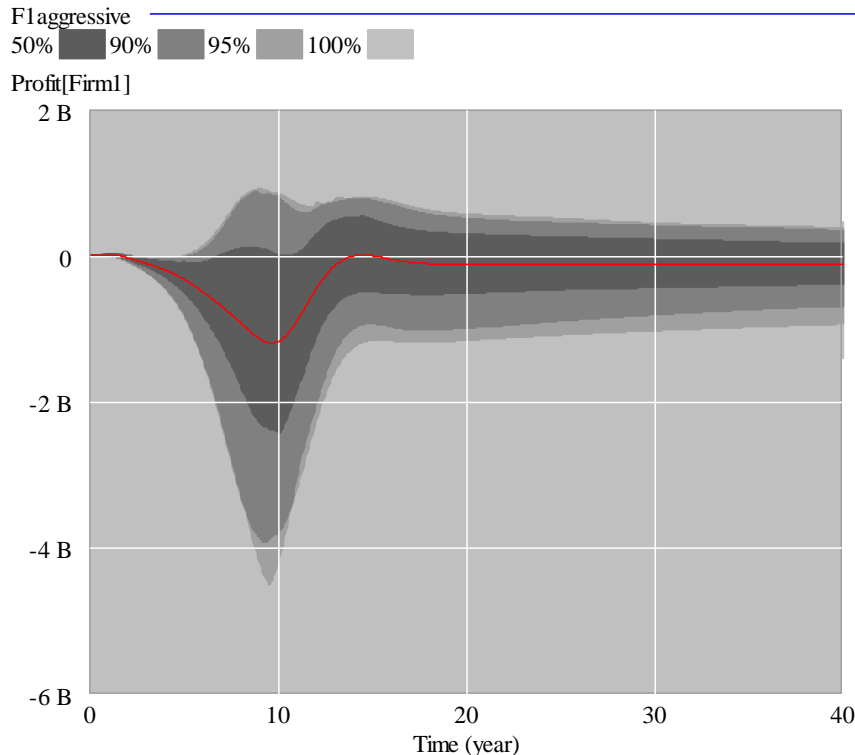
- Replace your best guess .out with the **Firm1aggressive.CIN** policy in the list of changes:



- Run a Sensitivity run.

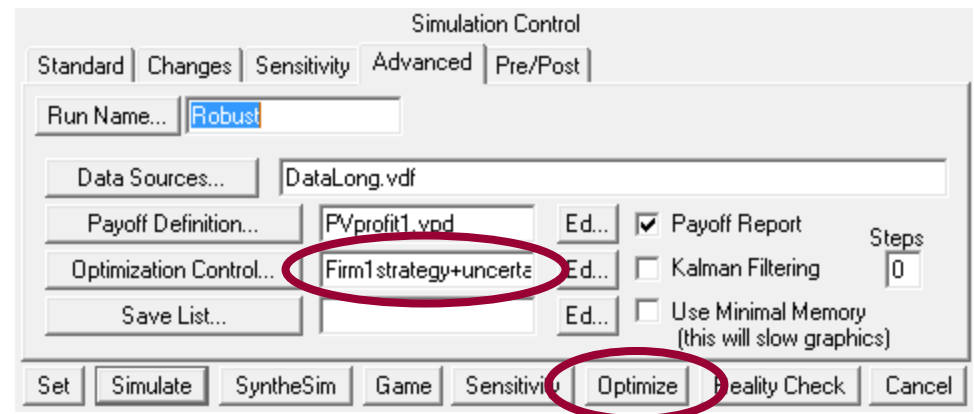
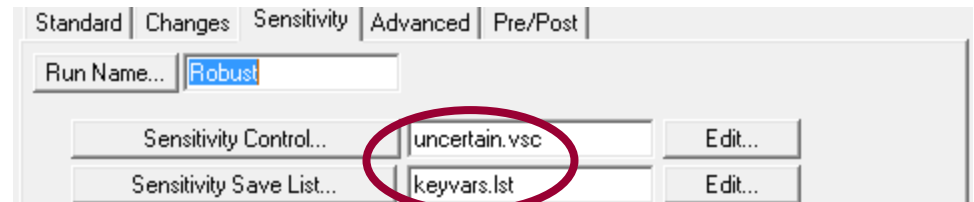
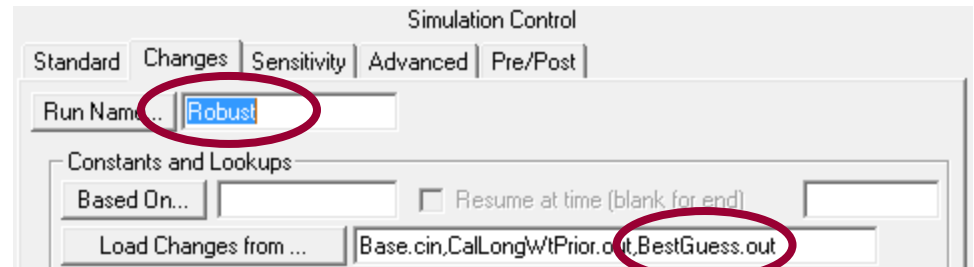
Results

- Use the Sensitivity Graph  and Histogram  tools to view distributions of outcomes (you may need to right-click tools to change settings, or load a new toolset)
- Explore key outputs:



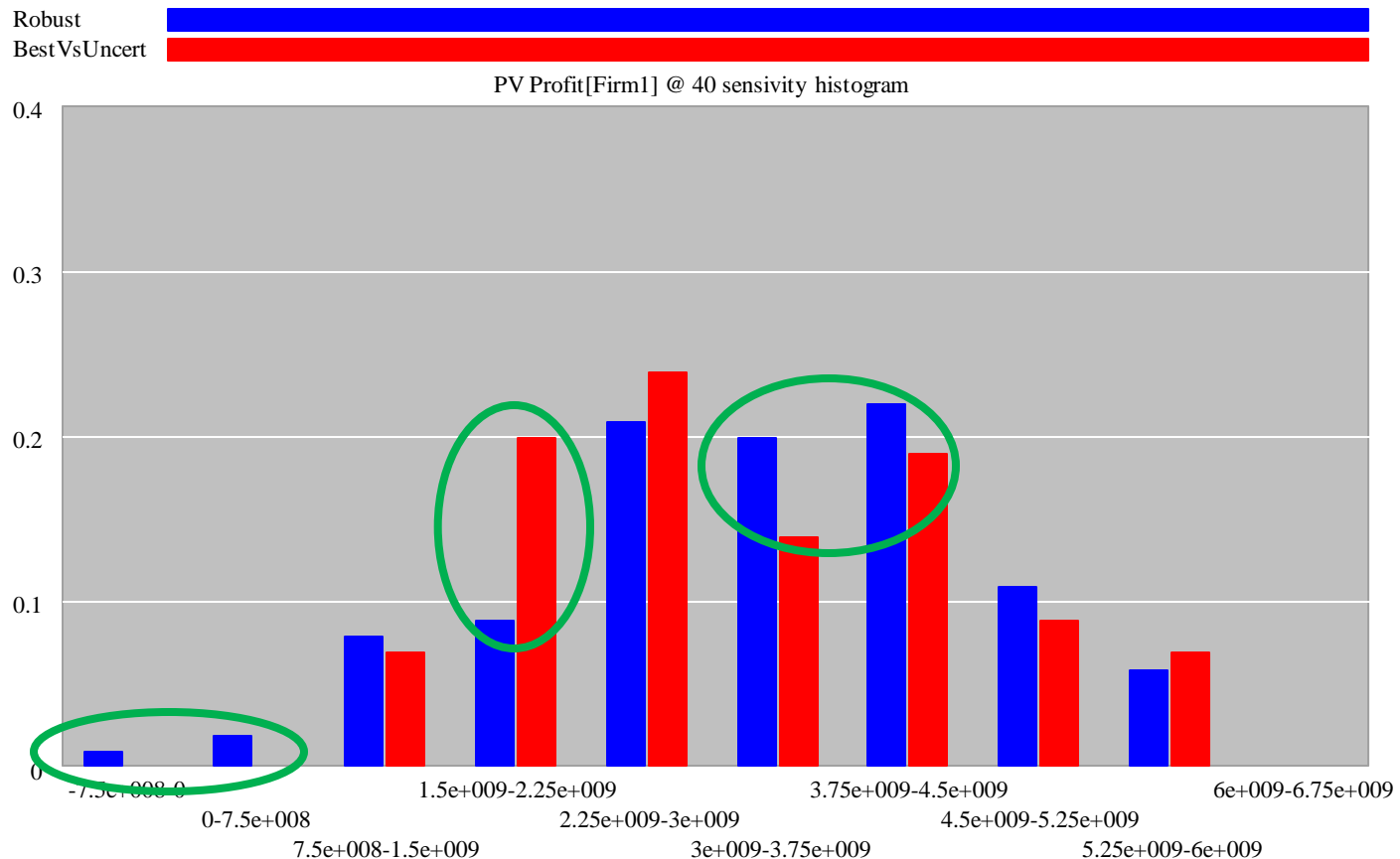
What's the Best Policy Under Uncertainty?

- **Stochastic Optimization:** hill climbing to maximize the sum of the outcomes of an ensemble of sensitivity runs
- Add the **:STOCHASTIC** keyword to optimization control file to combine optimization and sensitivity (Monte Carlo)
 - Use File>Edit... to inspect **Firm1strategy+uncertain.voc**
- Specify sensitivity files, then optimize



Result

- The robust policy trades some downside risk for slightly higher mean value



Is this what we want?

- **Maybe. First:**
 - Verify results with bigger samples
 - Improve the decision rules' structure
- **If not:**
 - Adjust the payoff (profit) to account for risk aversion more consistent with what we mean by “robust”

Note

- **The simulations in these slides can be replicated by running the two command scripts,**
 - Pt1 CalibExperiments.cmd, and
 - Pt2 PolicyExperiments.cmd
- **The last run in Pt1, MCMC, must be stopped manually (or it will run until disk space runs out)**