Model Calibration

Day 3

Model Calibration Overview

Motivation

- Mathematical models of disease often have a subset of parameters whose values are unknown
- Common reasons that the parameters' values are unknown include physical, feasibility, and ethical limitations
- Estimate values for these parameters by matching model outputs to observed outcomes
  - **We do this via Model Calibration**

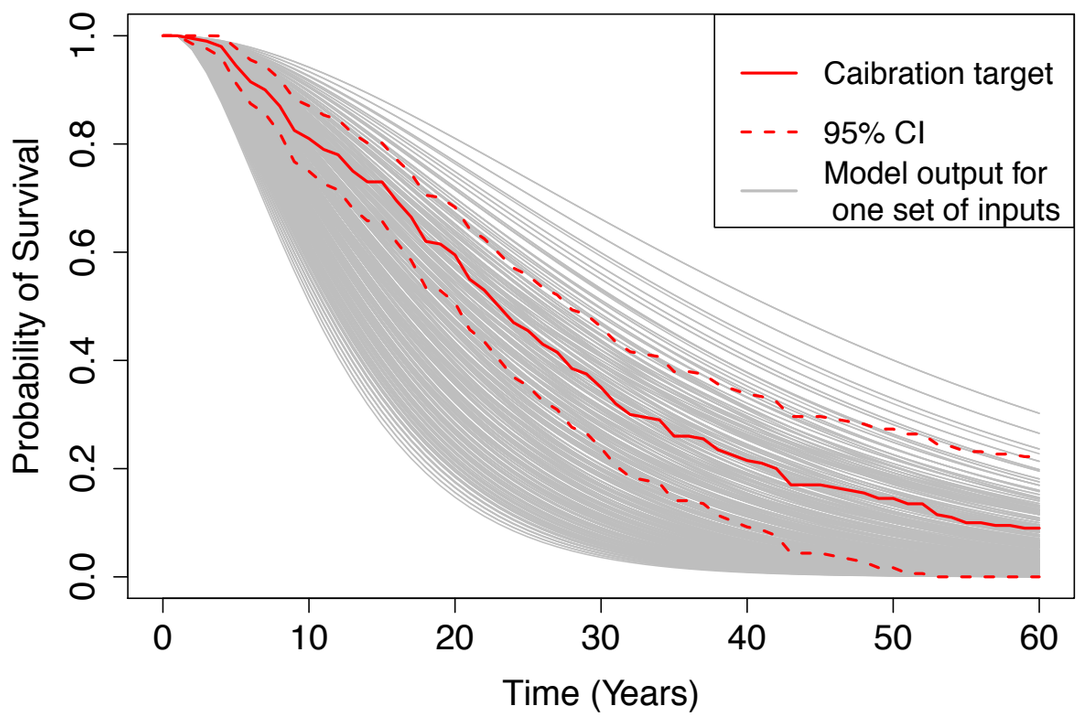
Calibration definition

- Model calibration involves the identification of input parameter values that produce model outputs that best predict observed data" (Karnon and Vanni (2011))
- Notation
  - M : a mathematical model (e.g., a Markov or microsimulation model)
  - θ : Set of K parameters to be calibrated
  - $\phi = M(\theta)$: Model output for parameter set θ
  - y : Values of T calibration targets

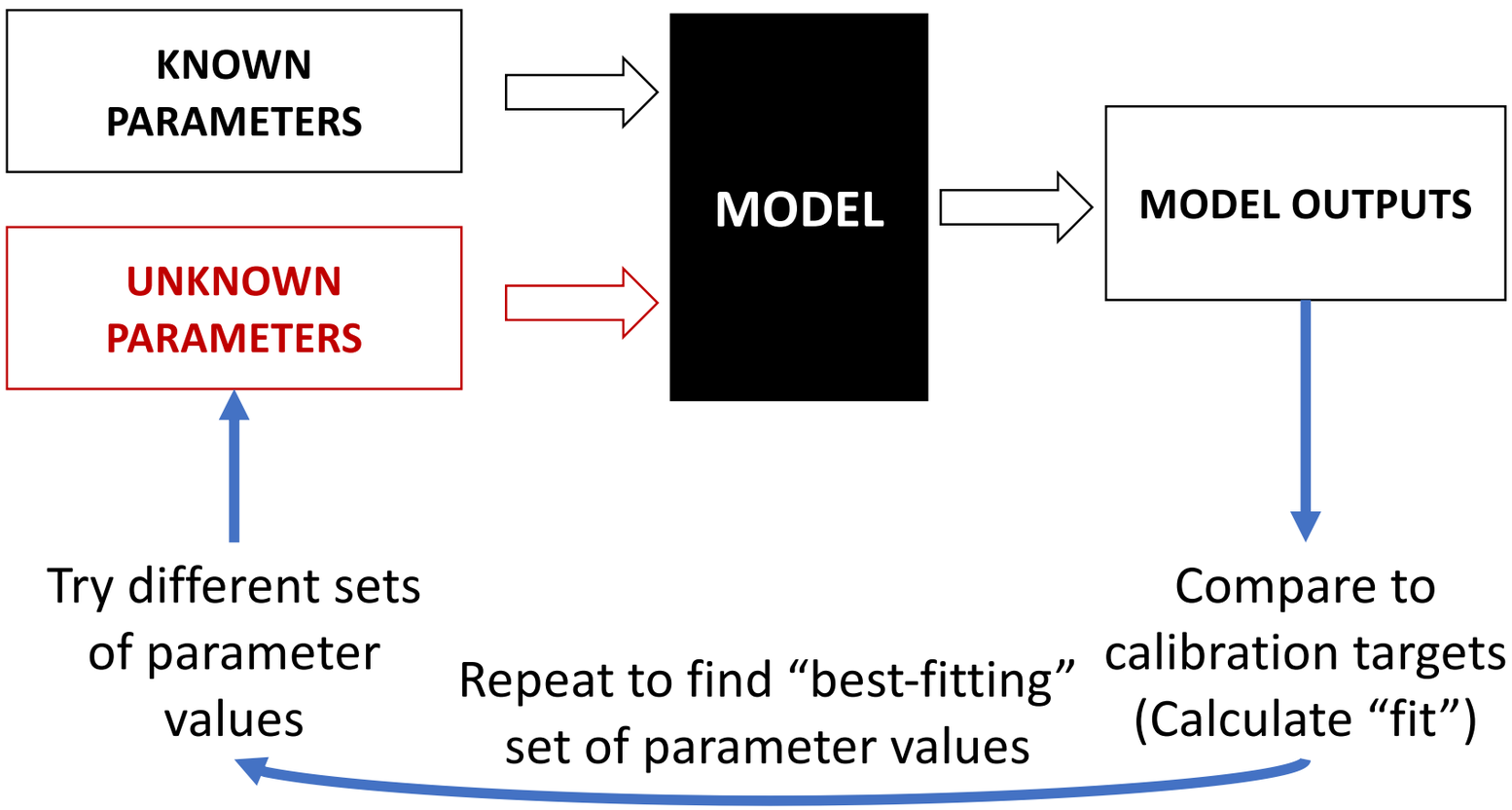
Calibration definition

- Process of estimating model input parameters to match data on outcomes of interest (e.g., survival, prevalence, or incidence)
- Outcomes of interest =

  “calibration targets”



Calibration process



Calibration components

- Target data = calibration targets
- Goodness-of-Fit (GoF) measures
- Search Strategies
- Acceptance Criterion(a)
- Stopping Rule
- Estimation approaches

Calibration targets

- Empirical data (observed) to be replicated by the model
- Data not used as direct model inputs
- Summary statistics (e.g. mean age of cancer diagnosis) or series of observations (e.g. age-specific incidence)
- Can calibrate to multiple targets (e.g. survival and prevalence) simultaneously
- Model should be able to output the outcomes of interest
    - Output coverage!

Goodness-of-fit (GoF)

Calculating “fit”

- Goodness-of-Fit (GoF) is the quantitative measure of how well the model is replicating the target data
- Different ways to measure GoF
  - Distance
  - Likelihood

Distance GoF measures

- Sum of squared errors

$$SSE(\theta) = \sum_{i=1}^{T} \left(y_i - M_i(\theta)\right)^2$$

- Weighted sum of squared errors
  - Assign different weight to different targets (w_i)
  - Often, $w_i = \frac{1}{\sigma^2}$

$$WSSE(\theta) = \sum_{i=1}^{T} w_i \left(y_i - M_i(\theta)\right)^2$$

Likelihood as GoF

- How likely is the observed data to have come from a model M with set of parameter values θ?
- Assuming targets are independent, overall likelihood is the product of individual likelihoods

$$L(y|M(\theta)) = \prod_{i=1}^{T} L_i(y_i|M_i(\theta))$$

- Generally, we work with log-likelihood

$$\mathcal{L}(y|M(\theta)) = \sum_{i=1}^{T} \log L_i\left(y_i|M_i(\theta)\right)$$

Likelihood as GoF

- In the case of Normal likelihood with equal variance across targets, MLE yields the same results as minimizing SSE. **But** if the variance is not equal, MLE is **NOT** equivalent to minimizing SSE
- The variance serves as a weighting factor. The lower the variance, the higher the weight! **Why?**

Commonly used likelihoods

- **Normal distribution**

$$\log L(y_i, \sigma_i | M_i(\theta)) = -\frac{1}{2}\log\left(2\pi\sigma_i^2\right) - \frac{1}{\sigma_i^2}\left(y_i - M_i(\theta)\right)^2$$

  - In R: `dnorm(x=`y_i`, mean=`$M_i(\theta)$`, sd=`σ_i`, log=T)`

- **Binomial distribution – for prevalence or proportions**

$$\log L(y_i, n_i | M_i(\theta)) = \log\left(\binom{n_i}{y_i}\right) + y_i \log\left(M_i(\theta)\right) + (n_i - y_i)\log\left(1 - M_i(\theta)\right)$$

  - In R: `dbinom(x=`y_i`, size=`n_i`, prob=`$M_i(\theta)$`, log=T)`

- **Multinomial distribution – for multiple categories proportions**
  - In R: `dmultinom(x=`y_i`, prob=`$M_i(\theta)$`, log=T)`

Commonly used likelihoods

- **Multivariate Normal (MVN) distribution – for correlated targets**

$$\log L(y, \Sigma | M(\theta)) = -\frac{1}{2}\left[\log(|\Sigma|) + \left(Y - M(\theta)\right)' \Sigma^{-1} \left(Y - M(\theta)\right) + k \log(2\pi)\right]$$

  - In R: `mvnfast::dmvn(x=`y`, mean=`$M(\theta)$`, sigma=`Σ`, log=T)`

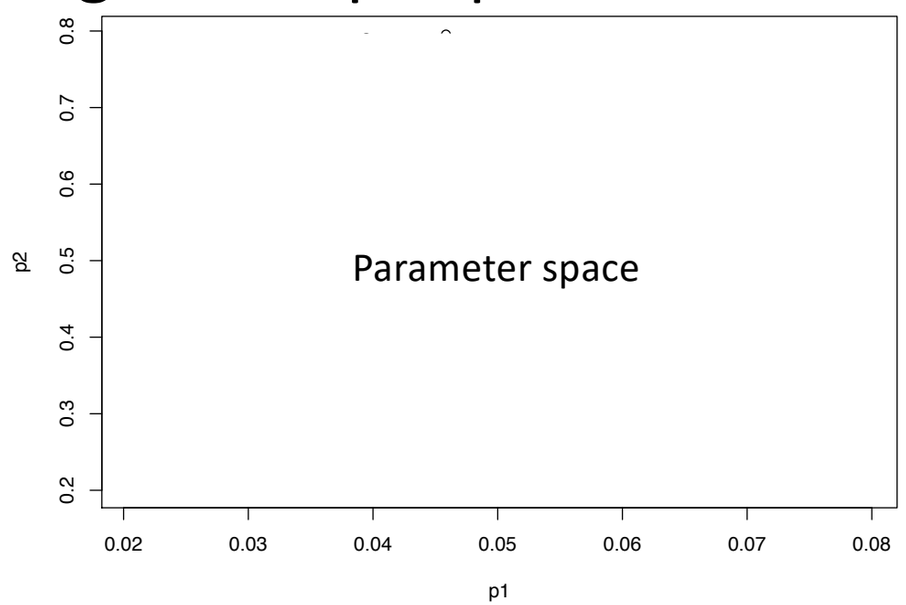
- **Poisson distribution – for counts, where y_i and $M_i(\theta)$ are cases**
  - In R: `dpois(x=`y_i`, lambda=`$M_i(\theta)$`, log=T)`

- **Poisson distribution – for incidence, where $M_i(\theta)$ is a rate and P_i is the population at risk**
  - In R: `dpois(x=`y_i`, lambda=`$M_i(\theta) * P_i$`, log=T)`

Search Strategies and Algorithms

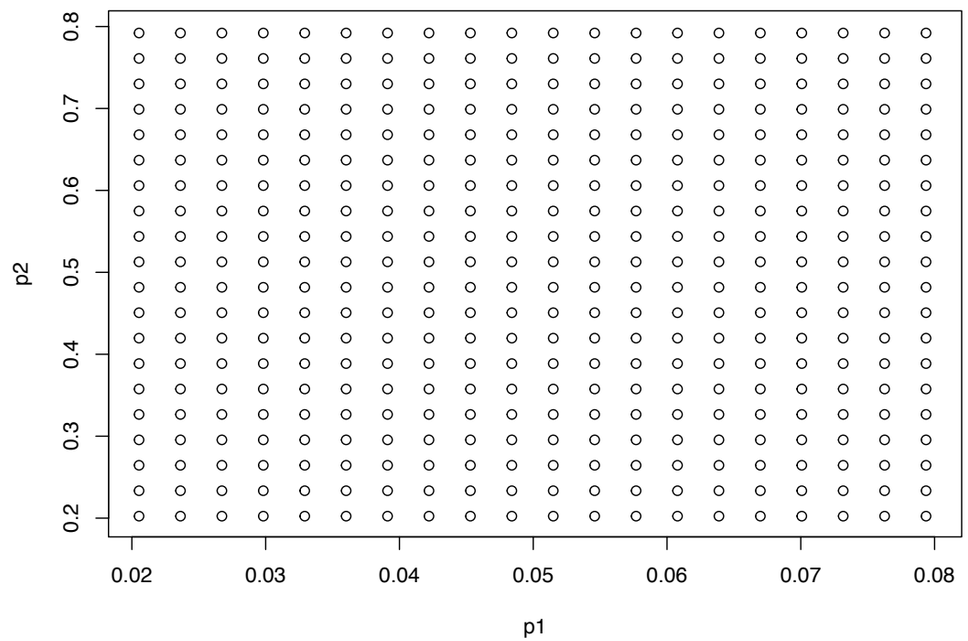
Search Strategy

- Define plausible ranges for parameter whose values are unknown
- Use a search strategy to “search” through the input parameter space
  - Run the model for sets of parameter values generated by search strategy



Grid Search

- Run model for all possible combinations of parameter values
- Often computationally infeasible



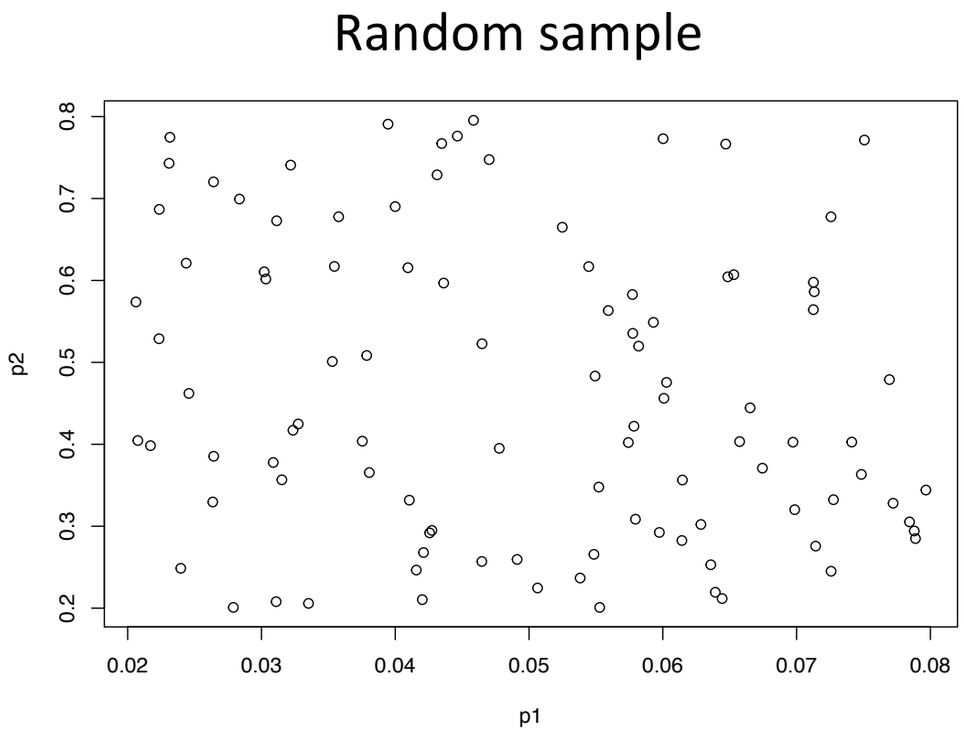
Random Search

- Randomly sample a large number of parameter value sets from probabilistic distributions
- We suggesting using

"Latin hypercube sampling”

(LHS) to ensure the sample

captures the full parameter space



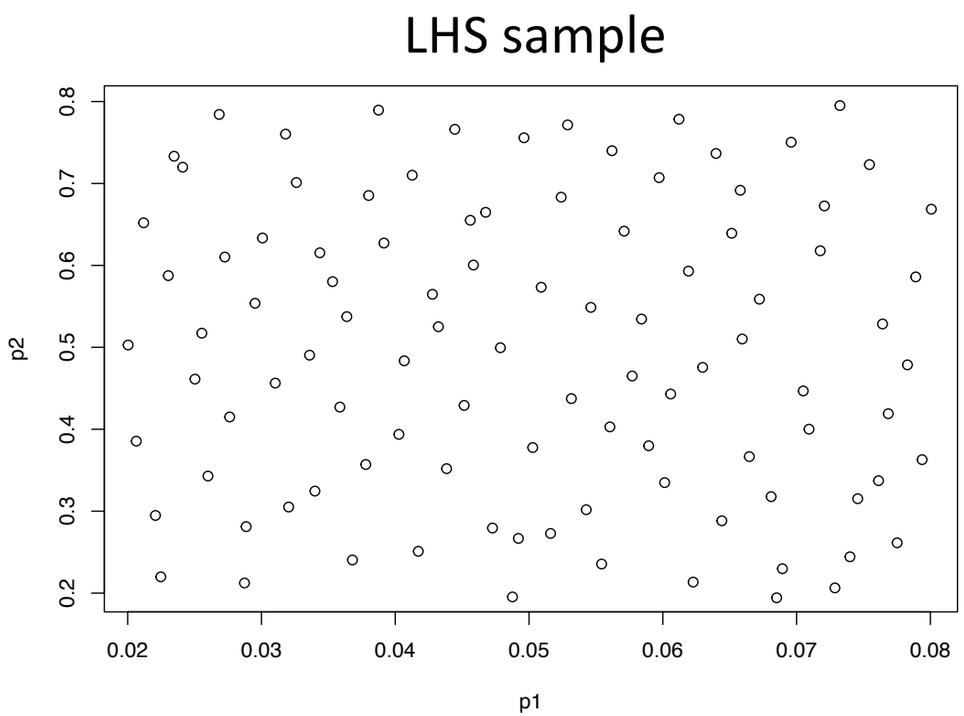
Random Search

- Randomly sample a large number of parameter value sets from probabilistic distributions
- We suggesting using

"Latin hypercube sampling”

(LHS) to ensure the sample

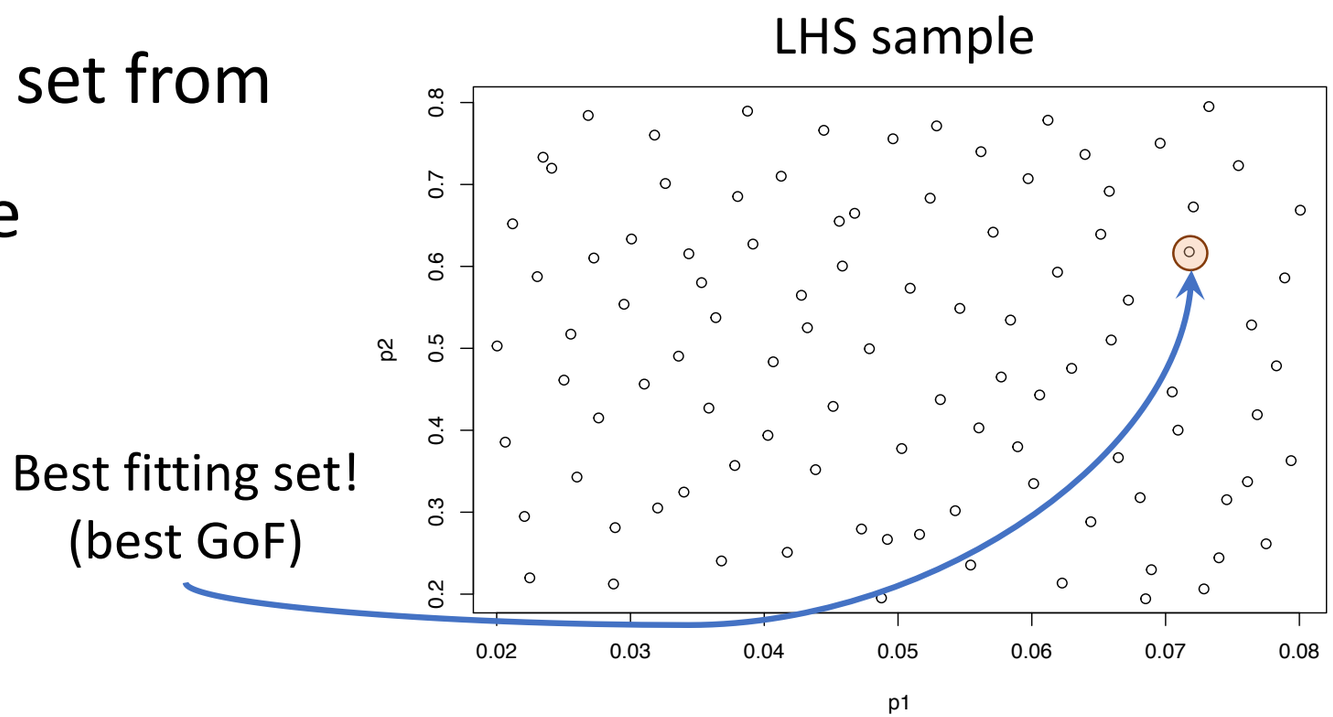
captures the full parameter space



Random Search

- Randomly sample a large number of parameter value sets from probabilistic distributions
- Identify the best fitting set from

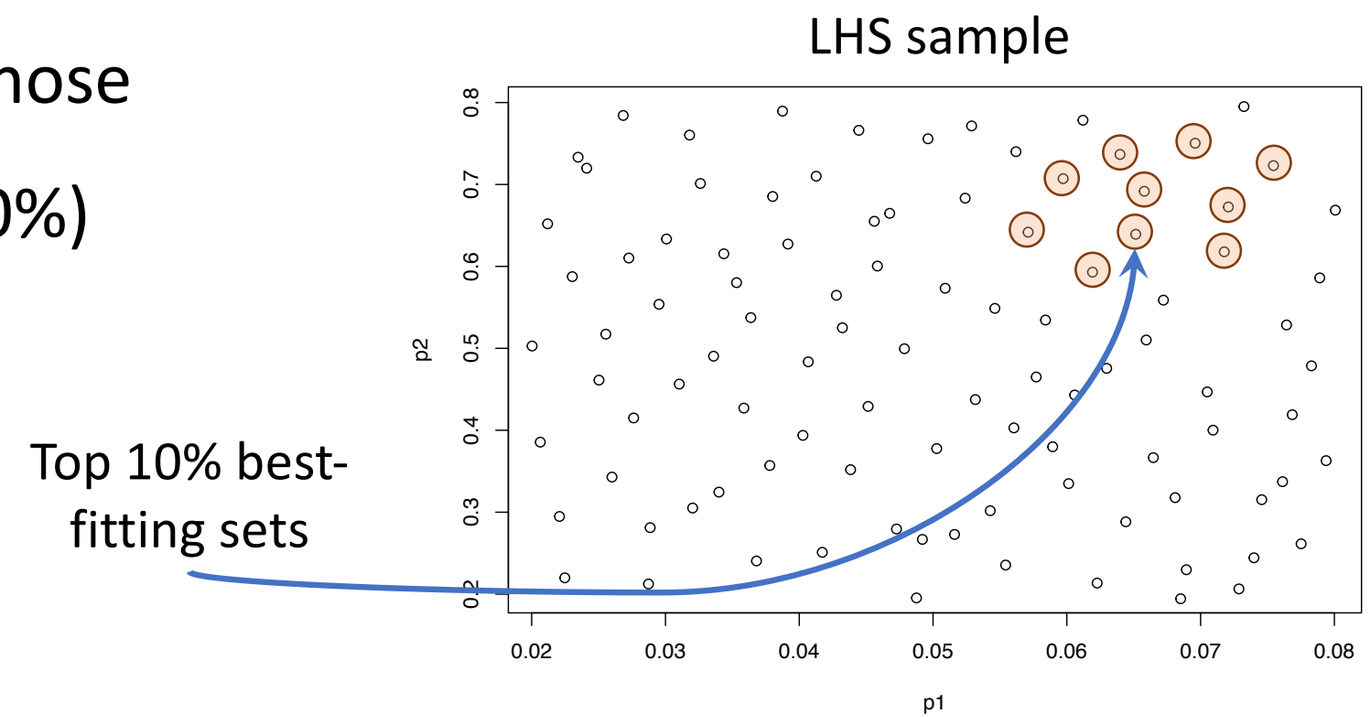
our large random sample



Random Search

- Randomly sample a large number of parameter value sets from probabilistic distributions
- Or identify the sets whose

fit is the top X% (e.g. 10%)



Iterative Search

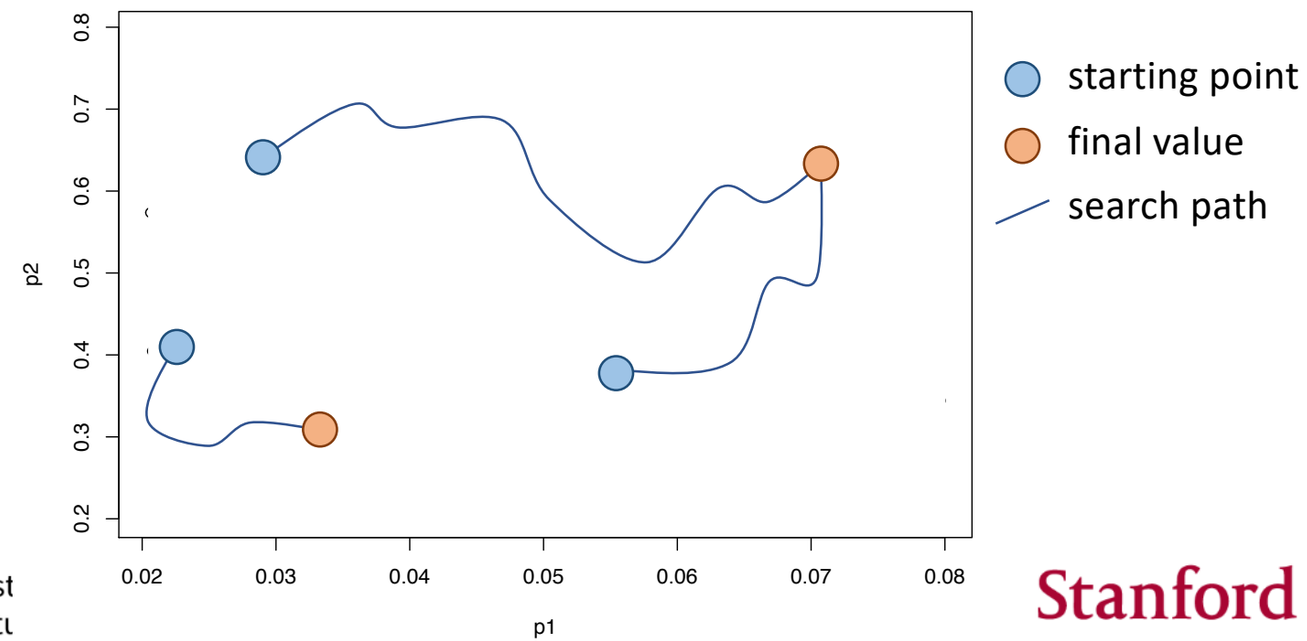
- Use fits of past input values to determine which input values to try next
- Directed methods
  - Nelder-Mead (simplex method)
  - Gradient–descent and others
- Meta-heuristic algorithms
  - Genetic algorithms
  - Simulated annealing

Nelder-Mead Algorithm

- Downhill simplex method
- It does not require the evaluation of derivatives like the gradient methods, only function evaluation
- **Pros:** Can be used to optimize non-differentiable functions (e.g., the simulation models we use!)

Nelder-Mead Algorithm

- **Cons**: Slower than gradient-based methods
- It can get "stuck" in a local critical point. Thus, the algorithm is usually run multiple times from different starting points



Uncertainty quantification

- Compute the Hessian matrix $\mathrm{H}(\theta)$ at best set (e.g., MLE)
- $\mathrm{H}(\theta)$ is a matrix of the second derivatives of the function → very useful!

$$\mathrm{H}(\theta) = \begin{pmatrix} \frac{\partial^2 f}{\partial \theta_1 \partial \theta_1} & \cdots & \frac{\partial^2 f}{\partial \theta_1 \partial \theta_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial \theta_n \partial \theta_1} & \cdots & \frac{\partial^2 f}{\partial \theta_n \partial \theta_n} \end{pmatrix}$$

Uncertainty quantification

- The covariance matrix of the estimated parameters, $\Sigma(\hat{\theta})$, can be estimated as

$$\Sigma(\hat{\theta}) = -\mathrm{H}(\hat{\theta})^{-1}$$

where $\hat{\theta}$ is the parameter estimate

- Draw a sample of parameters from a multivariate normal (MVN)

$$MVN\left(\hat{\theta}, \Sigma(\hat{\theta})\right)$$

Example:
Calibrating a 3-state cancer model

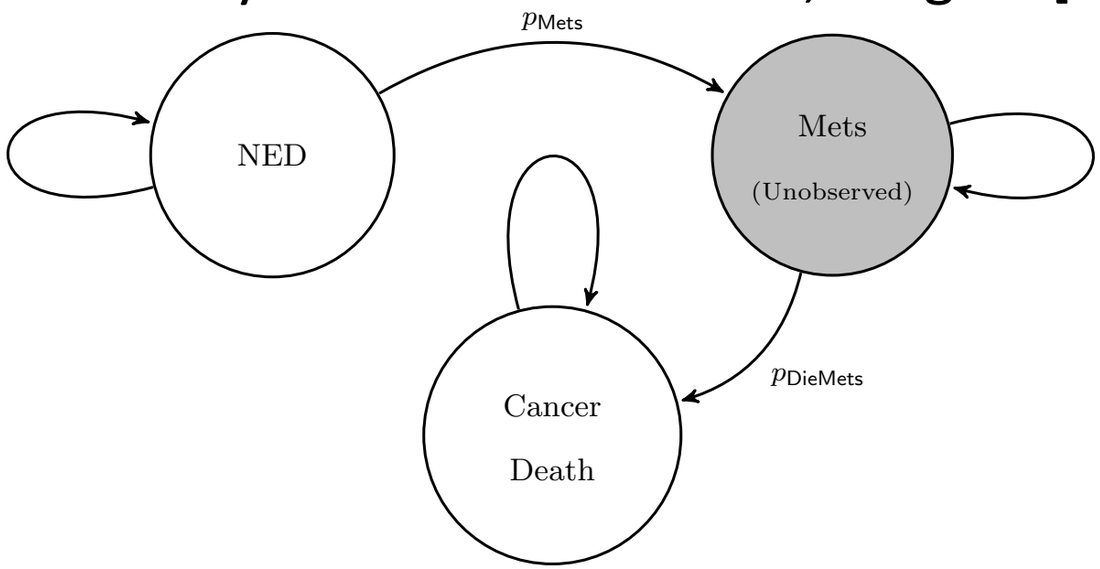
3-state cancer model

- Relative survival, as reported by the Surveillance, Epidemiology, and End Results (SEER) Program, represents cancer survival in the absence of other causes of death
- Cancer Markov models often have a distant metastasis state, a state not directly observed in SEER, from which cancer deaths are presumed to occur
- These models are used to model interventions that affect transitions to and from this state

Alarid-Escudero F, MacLehose RF, Peralta Y, Kuntz KM, & Enns, EA. Non-identifiability in model calibration and implications for medical decision making. *Medical Decision Making,* 2018;38(7):810–21.

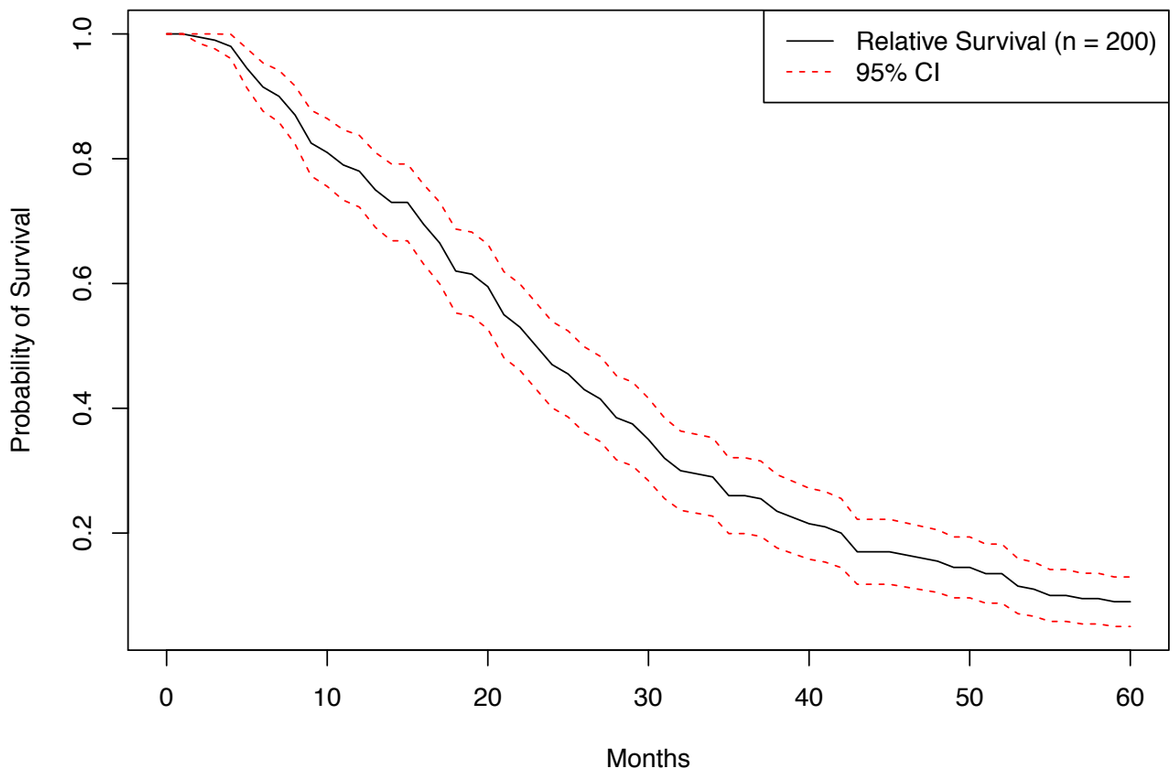
3-state cancer model

- Two unknown transition probabilities
  - p_Mets: Monthly risk of developing distant recurrence, range = [0.04, 0.16]
  - p_DieMets: Monthly risk of cancer death, range = [0.04, 0.16]



3-state cancer model

- Data frame “CRS_targets$Surv” stored in data file “CRS_CalibTargets.RData”



R Session

Bayesian Calibration

Bayesian setup

- In addition to the likelihood, we require a **prior**, which reflects previous belief or knowledge about the parameters of interest

$$p(\theta)$$

- Recall Bayes theorem

$$p(\theta|y) = \frac{p(y|\theta)p(\theta)}{p(y)}$$

  - $p(\theta)$ is the prior distribution
  - $p(y|\theta)$ has a "related" likelihood function $L(y|\theta) \propto p(y|\theta)$
  - $p(y) = \int p(y|\theta)p(\theta)\, d\theta$ is not a function of θ and often difficult to calculate

Bayesian setup

- Instead of a single best-fitting value, we would like an estimate of uncertainty in model parameters, θ, given our observed targets, y
  - E.g., a posterior distribution $p(\theta|y)$
- Notice that $p(\theta|y)$ is a distribution, not a single value.
- Relevant to our field. . . **Why?**

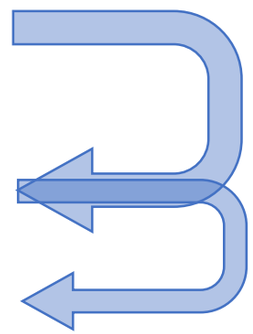
Bayesian setup

- A few steps of math...

$$p(\theta|y) = \frac{p(y|\theta)p(\theta)}{p(y)}$$

$$p(\theta|y) \propto p(y|\theta)p(\theta)$$

$$p(\theta|y) \propto L(y|\theta)p(\theta)$$



- Using this fact, we can sample the posterior distribution using the likelihood function, prior distribution, and some clever algorithms

Commonly used prior distributions to sample n_s values

- **Normal distribution**

  In R: `rnorm(n=`n_s`, mean=`$\bar{\theta}$`, sd=`σ_i`)`

- **Uniform distribution**

  In R: `runif(n=`n_s`, min=`lb`, max=`ub`)`

- **Beta distribution**

  In R: `rbeta(n=`n_s`, shape1=`α`, shape2=`β`)`

- **Gamma distribution**

  In R: `rgamma(n=`n_s`, shape=`α`, rate=`β`)`

Pros and cons

- **Pros:** We obtain distributions of parameters rather than only point estimates
- **Cons:** “Harder” to implement than other methods? Mmm not really!

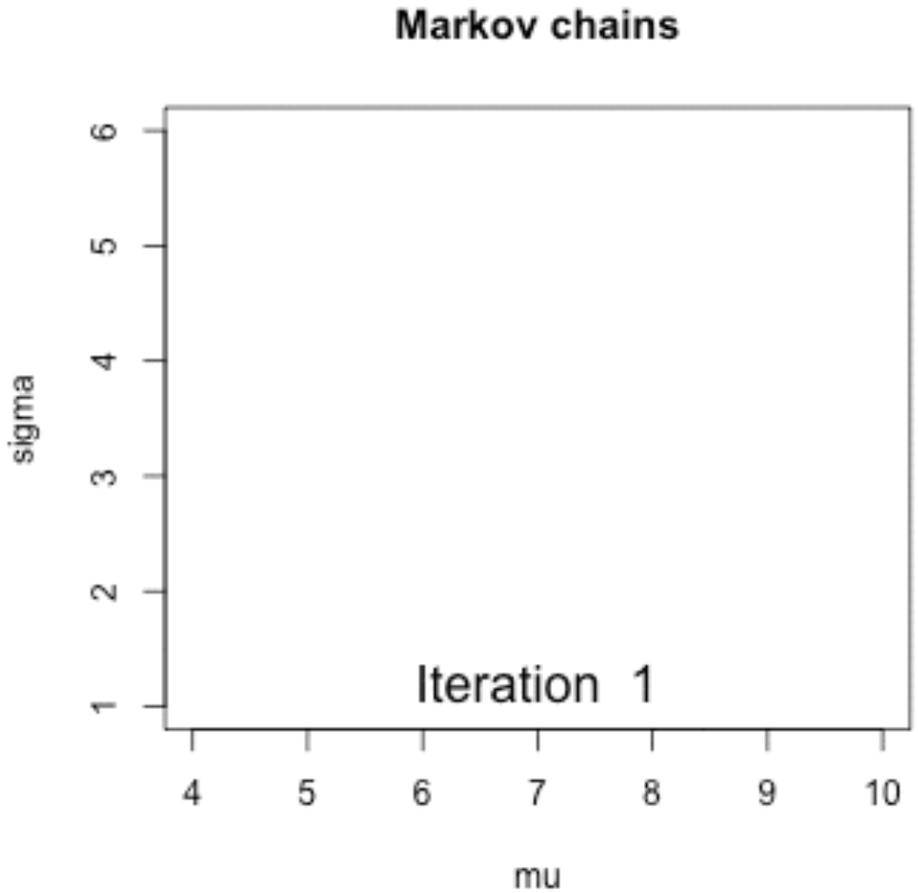
Obtaining a posterior distribution

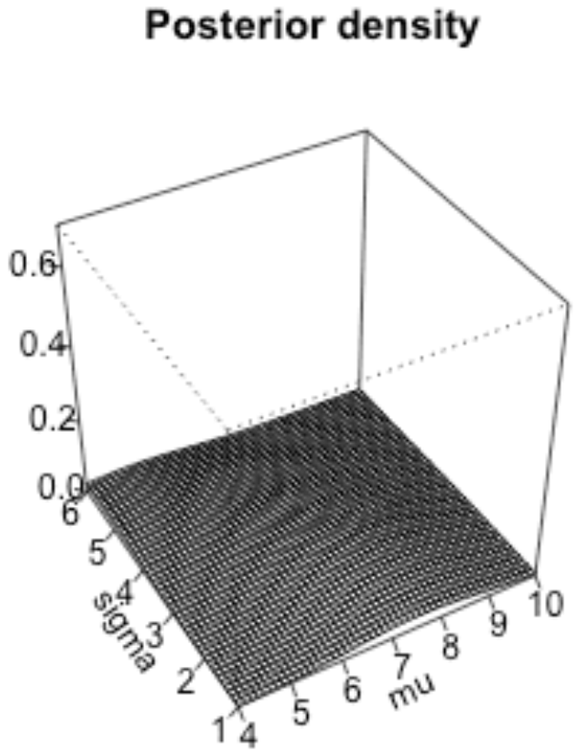
- Analytically
  - Not feasible for simulation models
- Markov chain Monte Carlo (MCMC)
- Resampling methods
  - Sampling Importance Resampling (SIR)
  - Incremental Mixture Importance Sampling (IMIS)

Markov chain Monte Carlo (MCMC)

- Simulates posterior distribution by **jumping** in the parameter space in a Markovian fashion
- The transition matrix of the Markov chain is obtained from a **proposal distribution**
- There are numerous algorithms that implement MCMC, including **Metropolis-Hastings** (MH)
- R has some packages that implement MCMC algorithms
- Other R packages interface with external MCMC software, including WinBUGS, JAGS, and stan

Markov chain Monte Carlo (MCMC)





Markov chain Monte Carlo (MCMC)

- **Pros:**
  - Theoretically will eventually converge to the “true” posterior distribution
- **Cons:**
  - Computationally intensive
  - Autocorrelation is a problem with simulation models and therefore requires a high number of samples
  - Use of external software (WinBUGS, JAGS, stan), requires implementing the model within that software’s syntax

Sampling Importance Resampling (SIR)

- Used to simulate posterior distributions (Rubin 1988) with three basic steps

1. **Sampling:** Sample a large number, N, of parameter sets from prior distributions
2. **Importance:**
    1. For each parameter set θ_i $(i = 1, \dots, N)$, run the simulation model and compute the likelihood
    2. Compute the (normalized) sampling importance weights w_i for each parameter set θ_i by dividing its likelihood value by the sum of likelihood values of all parameter sets,

$$w_i = \frac{L_i}{\sum_{i=1}^{N} L_i}$$

3. **Resampling:** Sample from the discrete distribution of $\{\theta(1), \dots \theta(N)\}$ with probabilities w_i

Incremental Mixture Importance Sampling (IMIS)

- SIR can miss areas of high posterior probability as the number of parameters increases
- IMIS addresses the limitations of SIR and was proposed by Steele et al. (2006)
- Starts with a modest-size SIR
- But in addition to SIR samples, add in samples from a mixture of multivariate normal distribution centered at the point with the highest importance weight
- Recalculate importance weights and sample a new sample of input parameter sets
- In the end, the posterior becomes a mixture of multivariate normal distributions and the prior distribution

Incremental Mixture Importance Sampling (IMIS)

1. **Initialization.**
   a) Sample N parameter sets from prior distribution
   b) For each parameter set, θ_i, calculate the likelihood L_i and compute the importance weights w_i
2. **Importance sampling**. Iterate for $k = 1, 2, \ldots$
   a) Choose a parameter set with maximum importance weight as the mean of a multivariate normal (MVN) distribution and compute the covariance matrix from nearby parameter sets. Define $H_k = \mathrm{MVN}(\theta^k,\ \Sigma^k)$.
   b) Sample B new input sets from H_k
   c) Re-calculate importance weights, w_i^k, for all $N + Bk$ samples
   d) Repeat Step 2 until the stopping criteria are met.
3. **Resampling**. Resample B_{re} input sets with replacement using importance weights from the last iteration, K. This is a sample from the posterior distribution

Incremental Mixture Importance Sampling (IMIS) Outputs

- `resamples:B.re` draws from the posterior distribution

- `stat`: diagnostic statistics at each IMIS iteration
  - `MargLike`:
  - `UniquePoint`: expected number of unique points among `resamples`
  - `MaxWeight`: maximum importance weight
  - `ESS`: effective sample size (the closer to `B.re`, the better)

- `center`: center of Gaussian components

R Session

Thank you!

Fernando Alarid-Escudero falarid@stanford.edu
Jeremy Goldhaber-Fiebert jeremygf@stanford.edu
Jorge Roa jorgeroa@stanford.edu

Stanford | Freeman Spogli Institute for International Studies

Stanford | Health Policy