SMC/Particle Filtering with Dynamic Models: A Spiral Tutorial

Nathaniel Osgood

University of Saskatchewan

Agenda

- Motivations & metaphors
- Particle filtering algorithm in a nutshell
- Case study
- Particle filtering: What
- Particle filtering: How
- Balancing model stochastics
- Particle filtering with agent-based models
- Summary
- Supplemental material

Slides (including supplemental material) available at:
https://tinyurl.com/PFMathEpi2019

Exemplar Context: Outbreak Prediction & Response

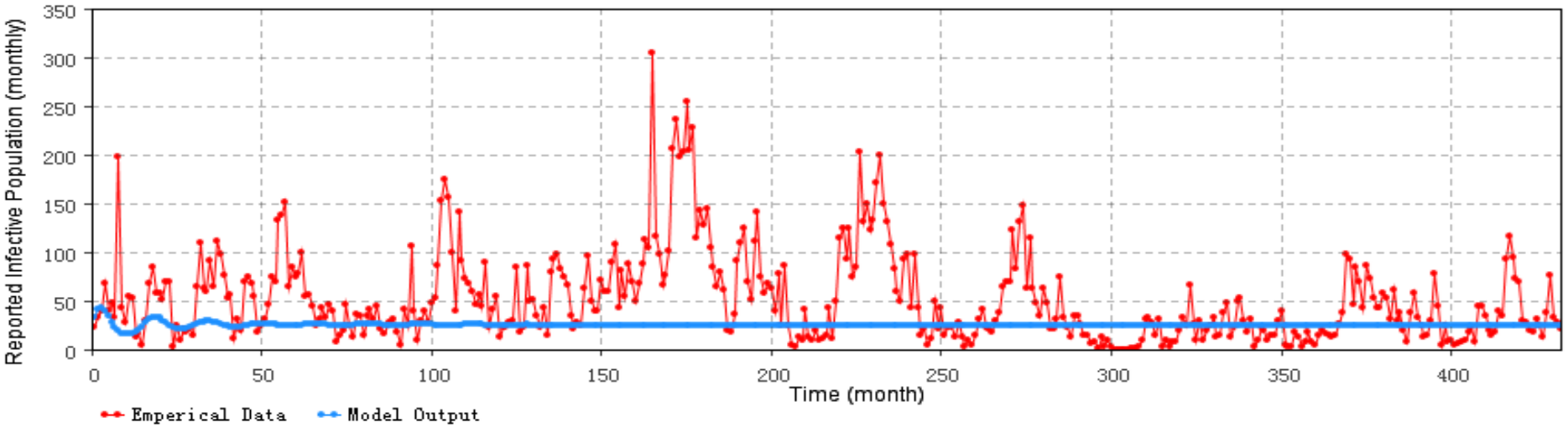
- Many concerns around mobilization of health resources hinge on outbreak detection and anticipation of evolution
- Regular (e.g., weekly) reporting often gives some sense as to “where we’re at”, but little clarity on what lies ahead
- Our focus lies in early detection of and anticipating trajectory of incident cases in an emerging outbreak
- Outbreak contexts are often marked by notable stochastics, including timing and evolution

Challenges for Dynamic Models

- With modeling projects, we typically
  - Build the model
    - Conceptualize, Formulate, Parameterize, Calibrate, Cross Validate & Otherwise test, …
  - Use it for insight
    - Running scenarios, studying outputs and intermediate dynamics, etc.
- After the model is formulated, often aligning it with new data is a heavy-weight ***manual*** process
  - Typical: Reparameterization, Recalibrating, Redoing cross-validation
  - Sometimes: Restratifying, reformulating, etc.

Problem B: Reflection 1

- Models as approximations: Unassisted, ***even the most detailed models*** eventually diverge from empirical situation as time passes
  - Divergence between model state & empirical state
- Some relevant challenges
  - **Stochastics**
  - Omitted factors
  - Approximations (e.g., random mixing)
  - New exogenous influences
  - Misunderstanding regarding process character
  - Latent heterogeneity
  - Mis-estimates

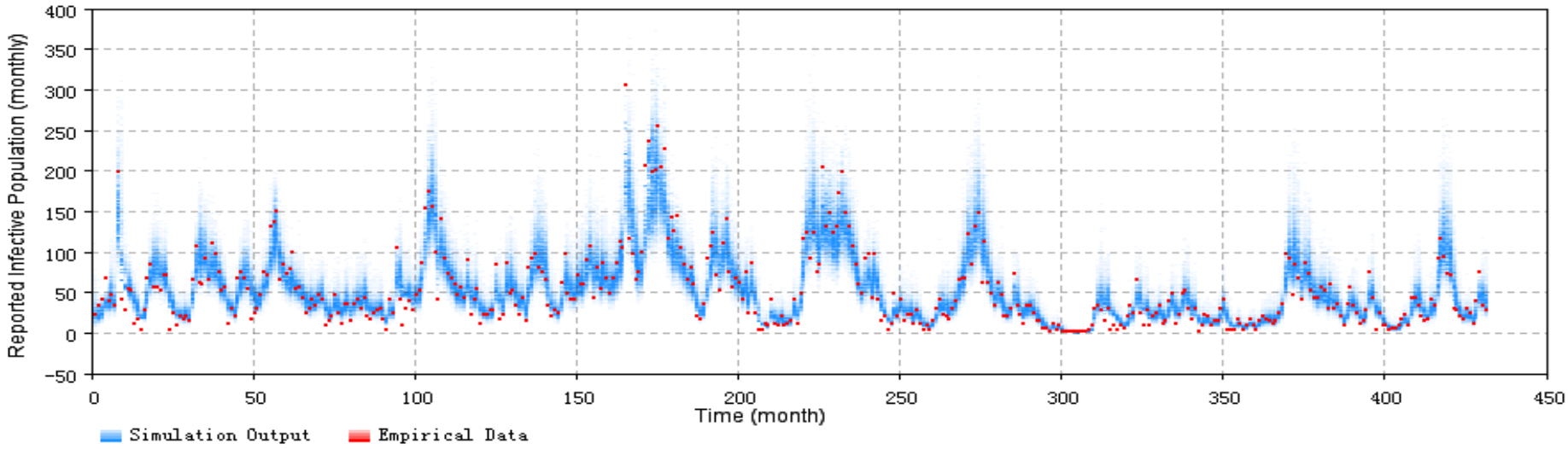

Reported Infective Population (monthly)
350
300
250
200
150
100
50
0
0
50
100
150
200
250
300
350
400
Time (month)
Emperical Data
Model Output

Problem B: Reflection 2

- While models can be very useful for responding to emerging issues, often there is little opportunity to craft a highly empirically grounded model
- Absent grounding in empirical evidence, accuracy of projections & analysis of policy tradeoffs is jeopardized
- By the time a highly grounded model can be created, often the greatest chance to shape policy is past

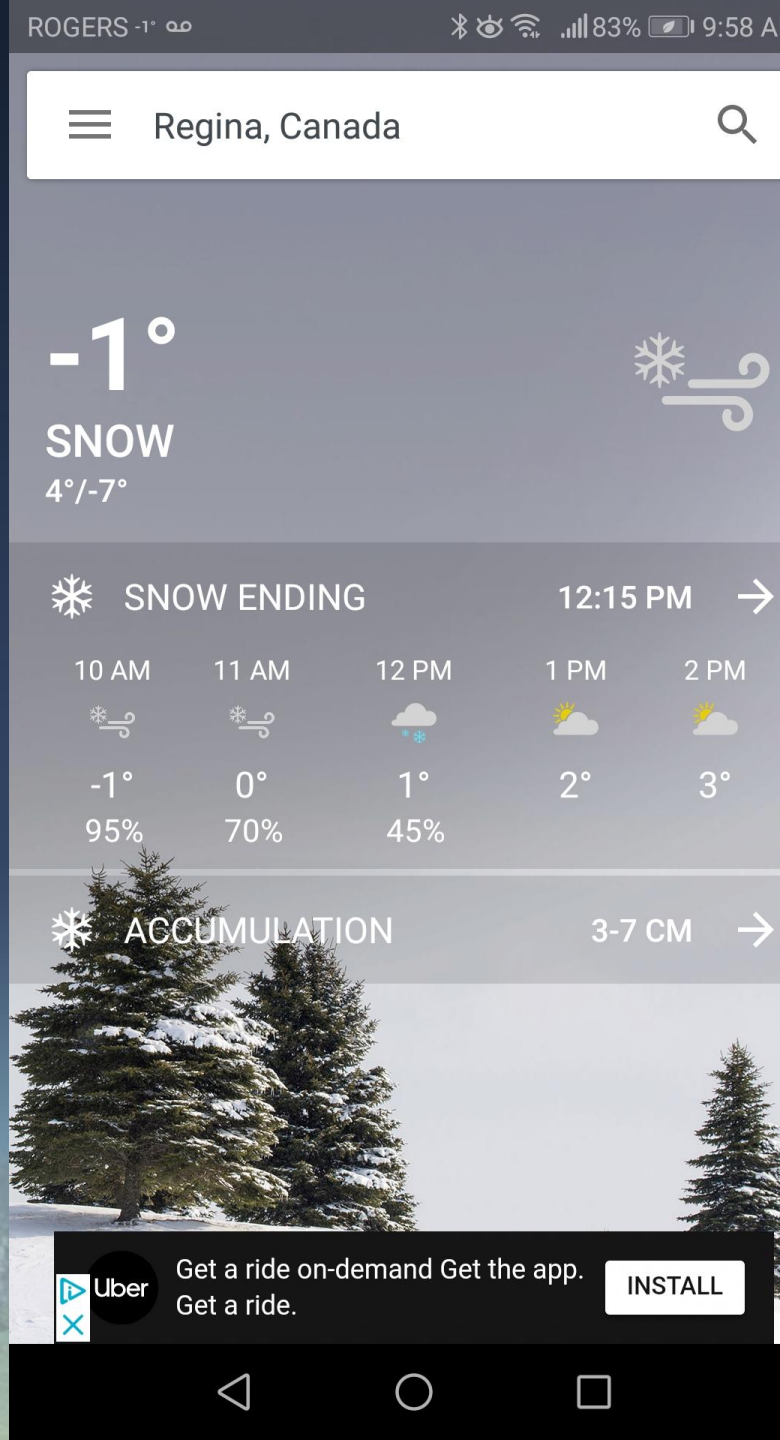
Solution Vision: Quickly Formulated, Frequently Regrounded Dynamic Models

- Rough & ready models quickly available to support decision-making, ***automatically*** regrounded & sharpened may be both more valuable and accurate than a far more detailed model that takes longer to create
- Rely not just on model predictions of where we're at, but also empirical observations that have unfolded
- Model state is kept "current" with the latest evidence, but can be used to project forward & study anticipated intervention tradeoffs
- Through logic of model structure together with time series, model regrounding illuminates latent areas of the system


Reported Infective Population (monthly)
400
350
300
250
200
150
100
50
0
−50
0
50
100
150
200
250
300
350
400
450
Time (month)
Simulation Output
Empirical Data

Avoiding Open-Loop Models

Like a weather report, we seek to always have the model updated to reflect the latest evidence & use that to anticipate future state (project forward)



Agenda

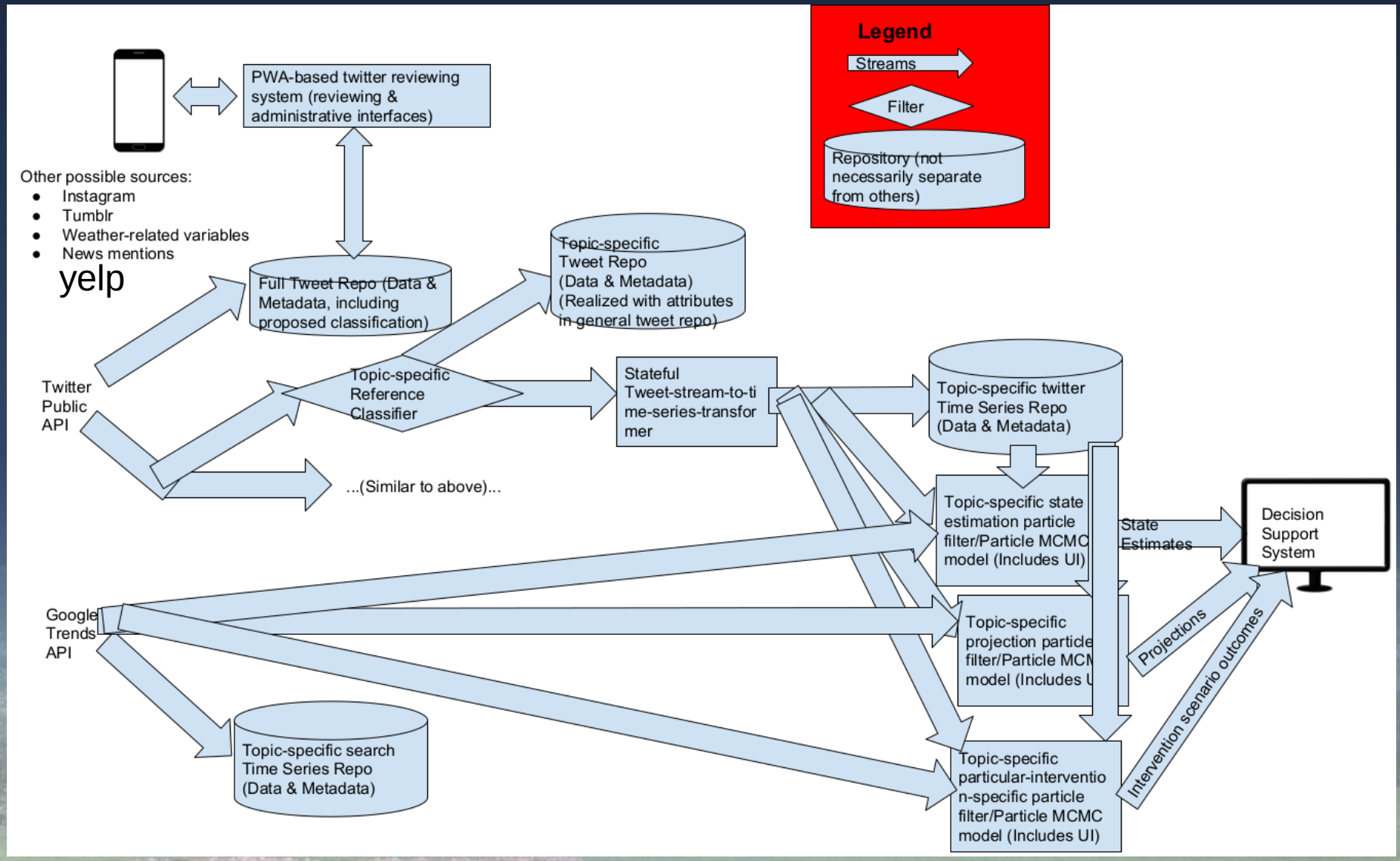
- ✓Motivations & metaphors
- Particle filtering “how” in a nutshell
- Case study
- Particle filtering: What
- Particle filtering: How
- Balancing model stochastics
- Particle filtering with agent-based models
- Summary

Our Past and Current Lines of PF/PMCMC Application

- Influenza[+]
- West Nile virus/Mosquito dynamics[+]
- Measles[+]
- Pertussis[+]
- Tuberculosis[+]
- Opioids[+]
- Suicide
- Chickenpox
- Emergency department waiting times
- HPA axis dynamics
- HIV immunity and HAART dynamics
- Criminal justice processes (e.g., remand, bail, probation, incarceration)
- H1N1 ground truth model testing

[+] indicates multiple explorations of particle filtered/PMCMC models

Context: Streaming, Repositories, Recurrently Regrounded Models for Decision Support



Machine Learning & Dynamic Models

- **MCMC: Sample from posteriors of *deterministic* dynamic model static parameters, latent states, scenario results, and incremental scenario gains.**
- **Particle Filtering/SMC: Sample from posteriors of *stochastic* dynamic model latent states, stochastically evolving parameters, scenario results, and incremental scenario gains.**
- **Particle MCMC (PMCMC): Sample from posteriors of *stochastic* dynamic model latent states, stochastically evolving parameters, scenario results, and incremental scenario gains *and* *static parameters*.**

Particle Filtering: A Few Key Facts

The simulation model

- *Only meaningful if include **stochastic processes***
- *Runs normally ("prediction") between observation points*
- *Is "corrected" to align w/empirical data at observation points*

Is performed ***recursively*** (in an ***on-line*** fashion)

- *Rather than re-estimate the state over all time points de novo when new data comes in, the estimate when new data comes in depends on estimates derived from earlier data*

Loose distributional assumptions (contrast: Kalman Filter)

No reliance on functional form/linearization of state equations

***Samples* from the state (and trajectory) distribution**

- ***Each sample is represented by a "particle"***
- *Particles reflect "competing hypotheses" as to the current state*
- *There is a "**survival of the fittest**" of particles (hypotheses)*
- *Exploits **importance sampling**: distribution is sampled by associating samples from proposal dist (particles) w/**weights***

How to Perform PF on ODE in a Nutshell

Start with stochastic ODE model

Subscript ODE model by 100s to 1000s of particles

Each particle has its own full copy of model state (anything that could differ b/t realizations)

Sample from initial model state from prior distribution; set weights uniformly to 1

(Prediction phase): Between observations

***All** particles evolve according to standard model dynamics (just perform integration of each particle's state until next observation; all particles survive)*

Particle weights remain invariant

(Update phase): At observation points: For each particle, **multiply particle weight by likelihood of observing the empirical observation vector, given particle state**

Resampling/"Survival of the fittest": If effective sample size is too low (too much disparity in weights) following observation, particles are resampled according to their weights, and weight is reset to 1

Particles with high weights reproduce; with low weights disappear

Trajectories can be sampled by maintaining ***ancestry matrix** holding lineages*

Agenda

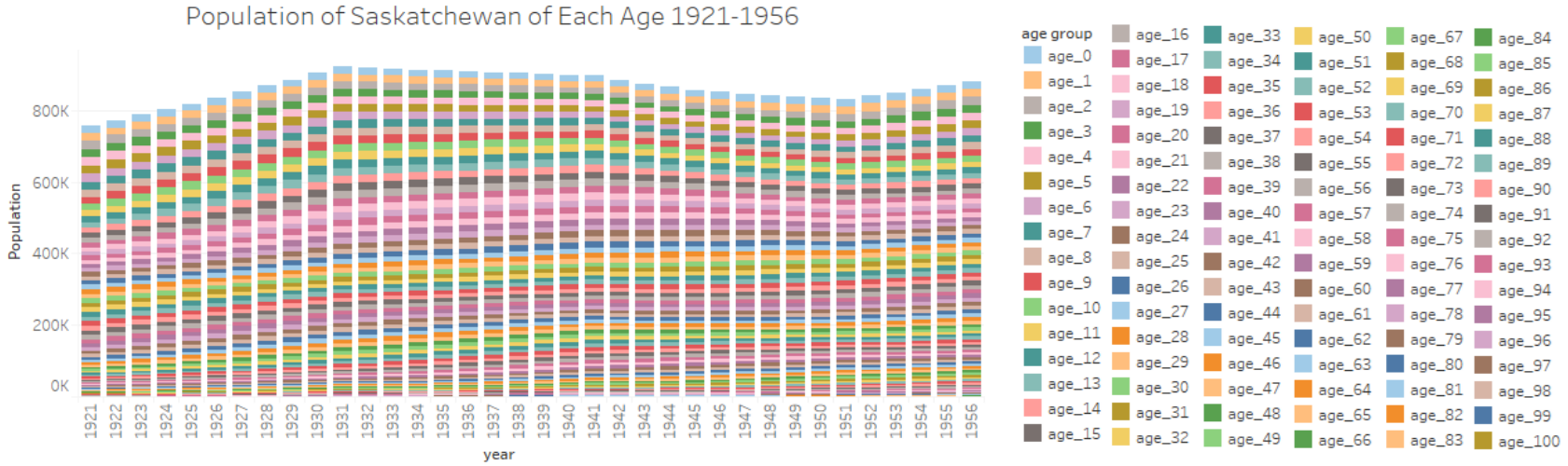
- ✓Motivations & metaphors
- ✓Particle filtering “how” in a nutshell
- Particle filtering: What
- Particle filtering: How
- Balancing model stochastics
- Particle filtering with agent-based models
- Summary

Agenda

- ✓Motivations & metaphors
- ✓Particle filtering “how” in a nutshell
- Case study
- Particle filtering: What
- Particle filtering: How
- Balancing model stochastics
- Particle filtering with agent-based models
- Summary

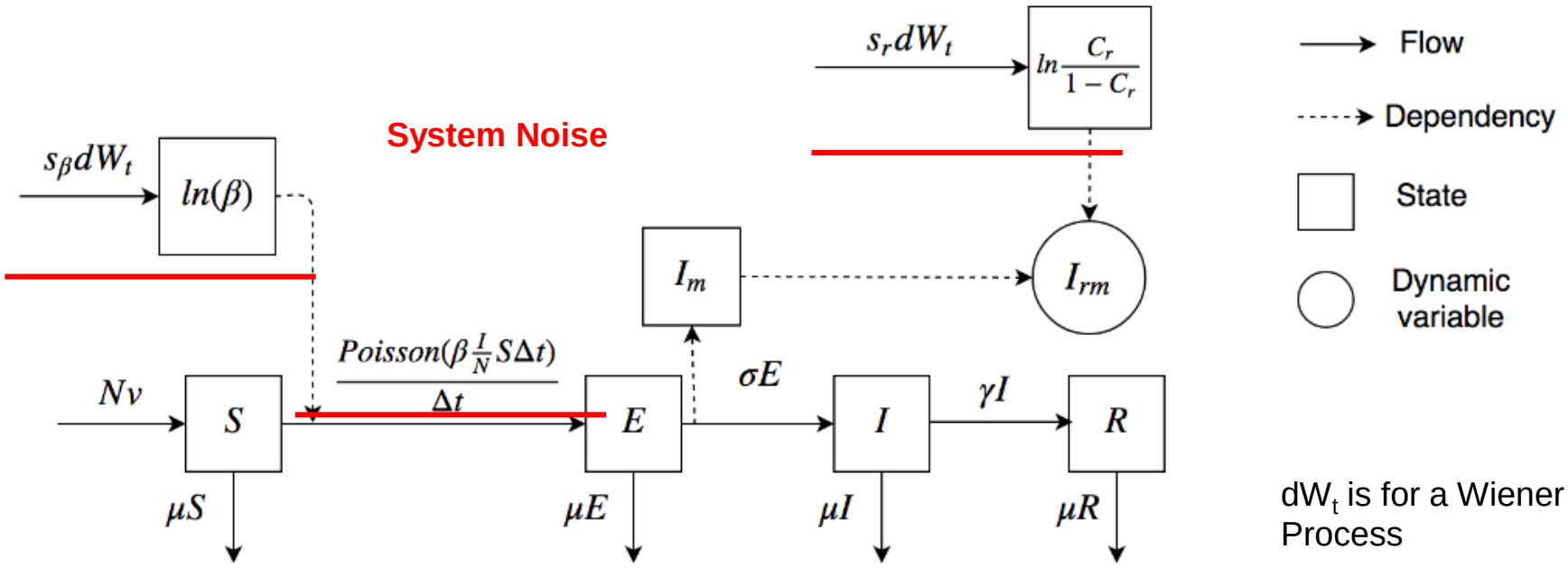
Empirical Datasets

- Measles and pertussis reported cases of Saskatchewan in pre-vaccination era (1921-1956):
  - Monthly reported cases across all population
  - Yearly reported cases in different (6) age groups
- Demographic data of Saskatchewan from 1921 to 1956.



Measles particle filtering models

Structure 1

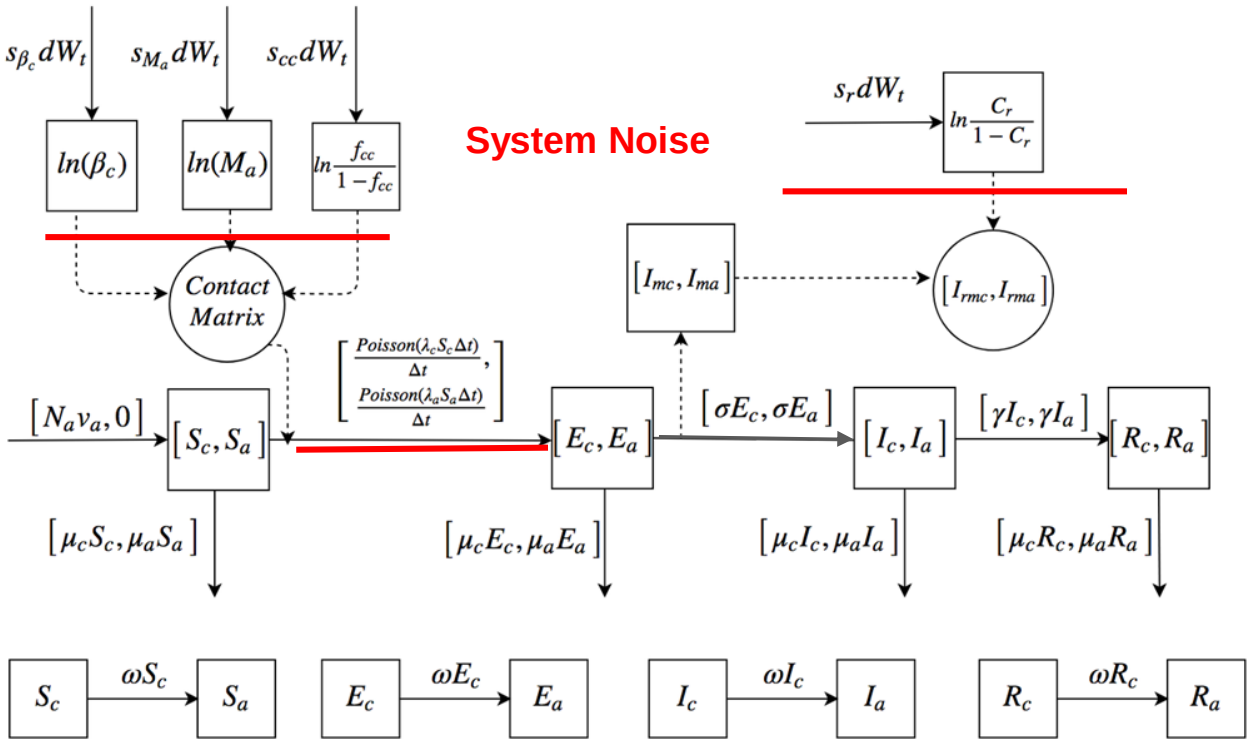


dW_t is for a Wiener Process

Structure of measles aggregate population particle filtering model

Measles particle filtering models

Structure 2



Flow

Dependency

State

Dynamic variable

$$Contact\ Matrix = \begin{bmatrix} \beta_c f_{cc} & \beta_c f_{ca} \\ M_a \beta_c f_{ac} & M_a \beta_c f_{aa} \end{bmatrix}$$

$$\begin{bmatrix} \lambda_c \\ \lambda_a \end{bmatrix} = \begin{bmatrix} \beta_c f_{cc} & \beta_c f_{ca} \\ M_a \beta_c f_{ac} & M_a \beta_c f_{aa} \end{bmatrix} \times \begin{bmatrix} \frac{I_c}{N_c} \\ \frac{I_a}{N_a} \end{bmatrix}$$

dW_t is for a Wiener Process

Structure of measles age-structured population particle filtering models (2 age groups)

Results for Measles

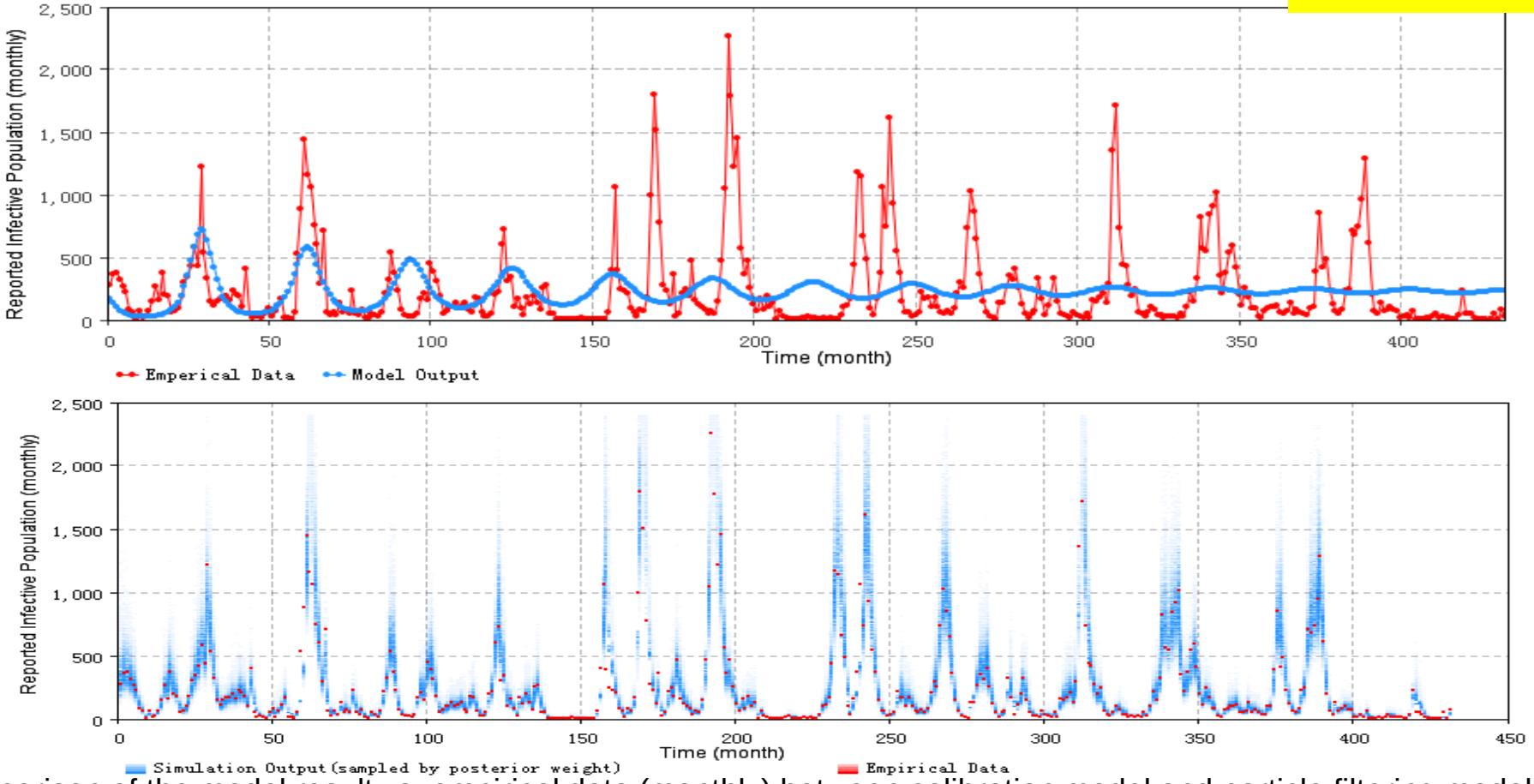
Comparison of the average RMSE of all models by incorporating empirical data across all observation points

Model	Monthly	Yearly in Month	Total
$Pure_{aggregate}$	249.0	NONE	NONE
$Calibrated_{aggregate}$	207.5	NONE	NONE
$PF_{aggregate}$	104.6 (99.4, 109.9)	NONE	NONE
$PF_{age\_5\_monthly}$	96.1 (91.5, 100.7)	179.5 (160.6, 198.3)	275.5 (260.2, 290.9)
$PF_{age\_5\_both}$	97.8 (94.1, 101.5)	144.8 (112.5, 177.1)	242.6 (210.2, 275.1)
$PF_{age\_15\_monthly}$	95.7 (89.8, 101.6)	45.9 (39.9, 51.9)	141.6 (133.8, 149.4)
$PF_{age\_15\_both}$	96.1 (91.6, 100.5)	39.9 (34.3, 45.6)	136.0 (127.6, 144.5)

- The sampled discrepancy of model's estimations vs. observed data is reduced by a factor of 2.0
- The age-structured model (PFage_15_both) is the minimum discrepancy model.

Estimation Results of Measles

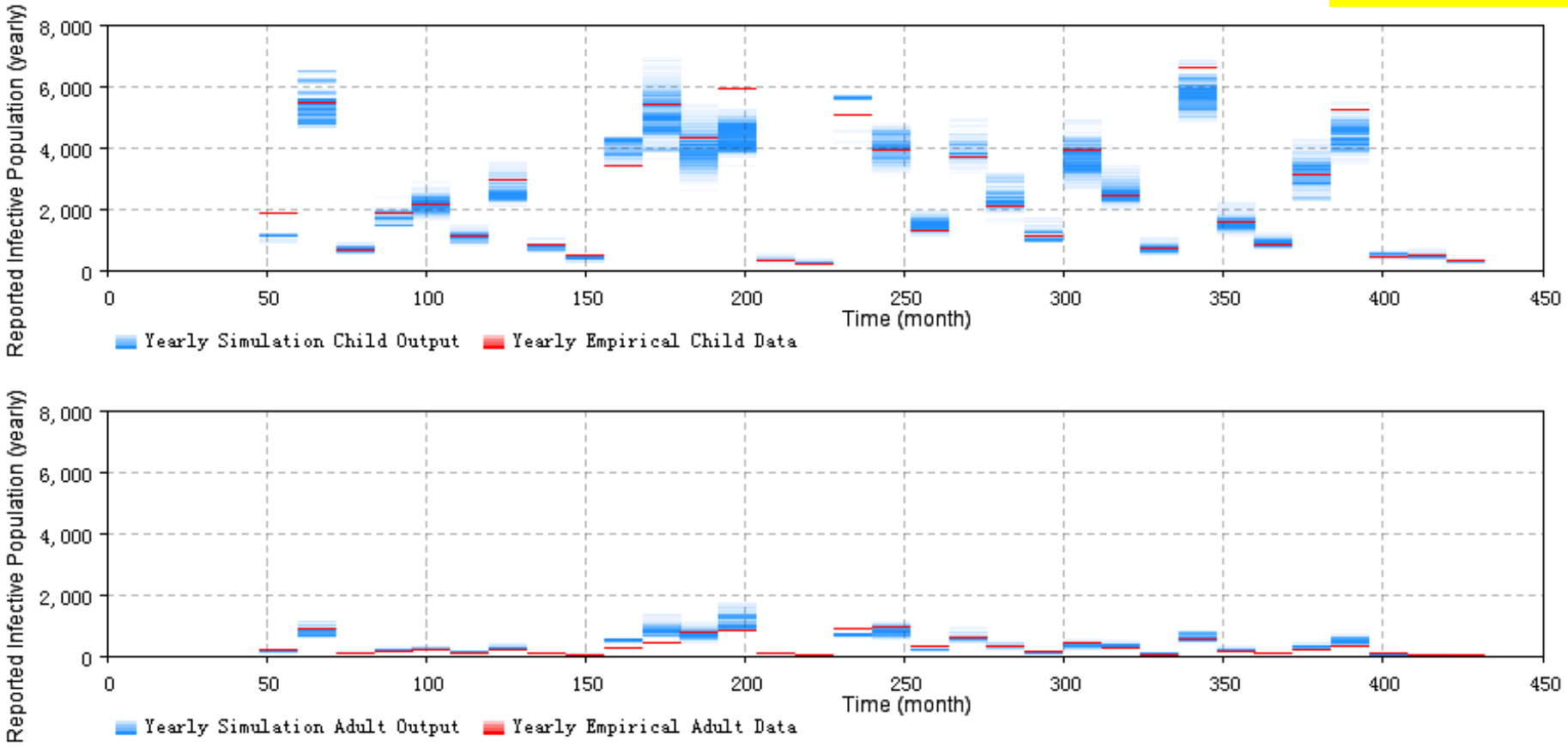
measles



Comparison of the model result vs. empirical data (monthly) between calibration model and particle filtering model

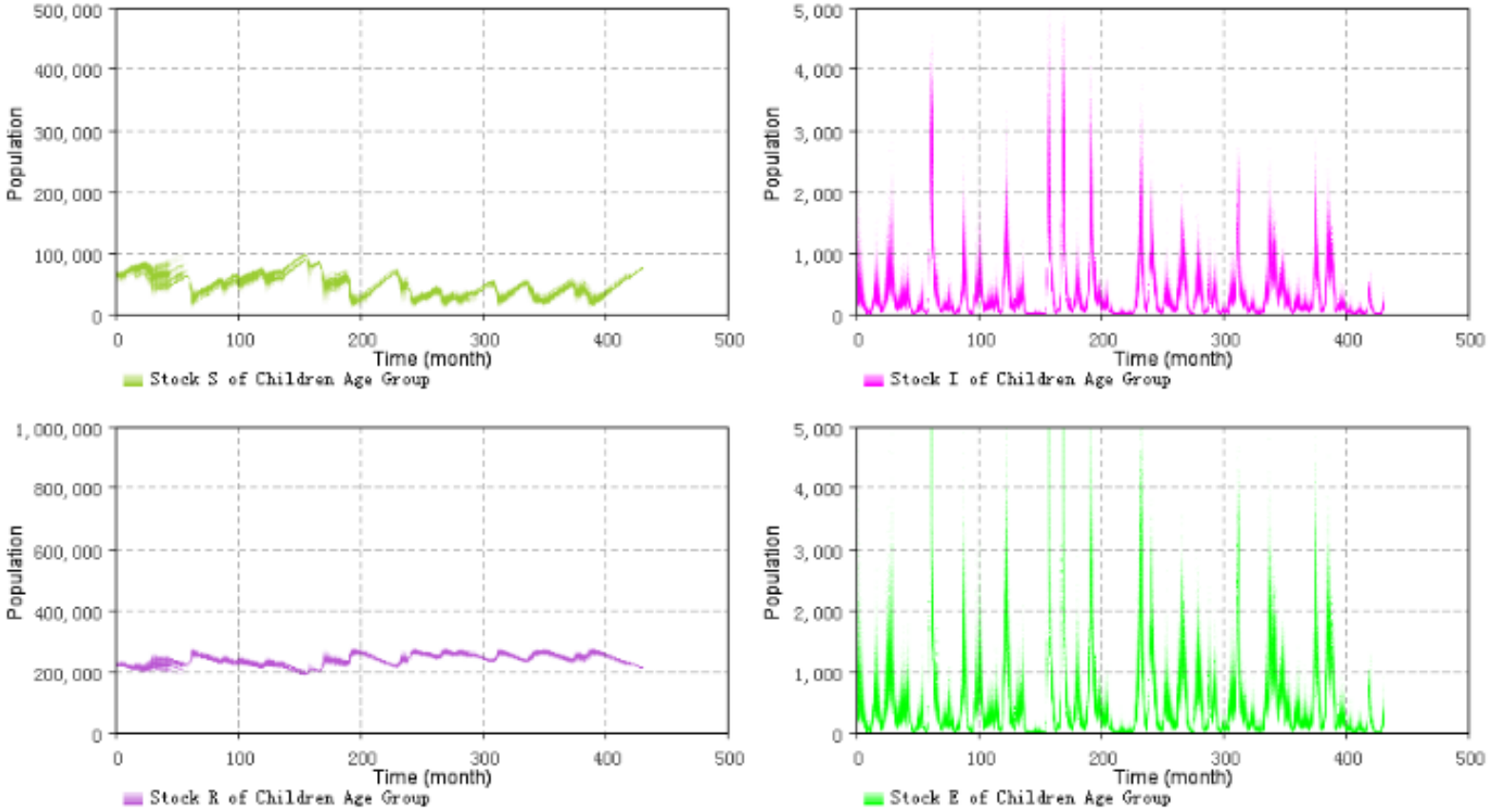
Estimation Results of Measles

measles



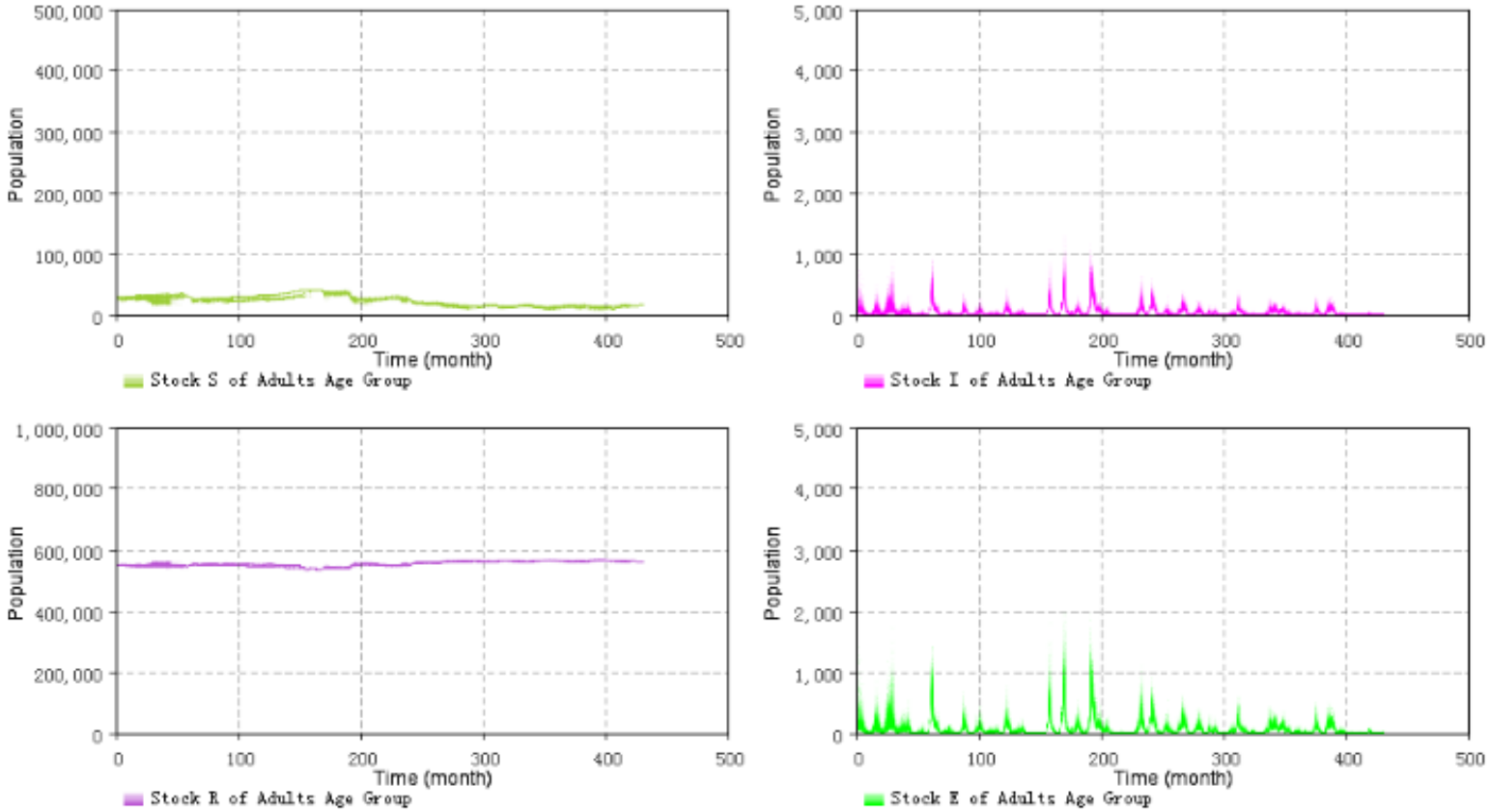
model result vs. empirical data (monthly)

Latent State for Measles - Children



The estimation of latent state (S, E, I, R) with the Child age group (less than 15 years) .

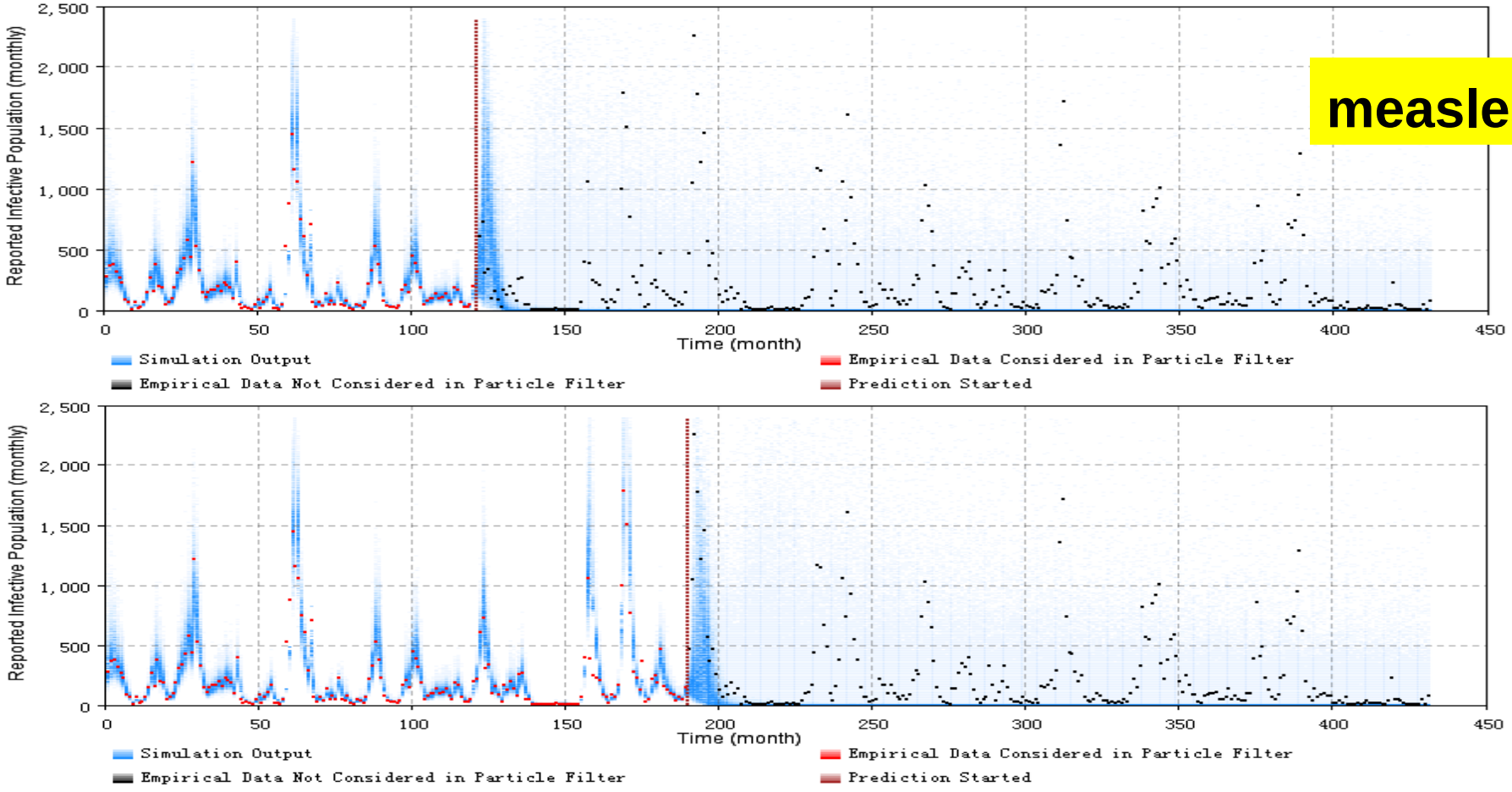
Latent State for Measles - Adults



The estimation of latent state (S, E, I, R) with the Adult age group (equal and greater than 15 years) .

Prediction results of the minimal discrepancy model

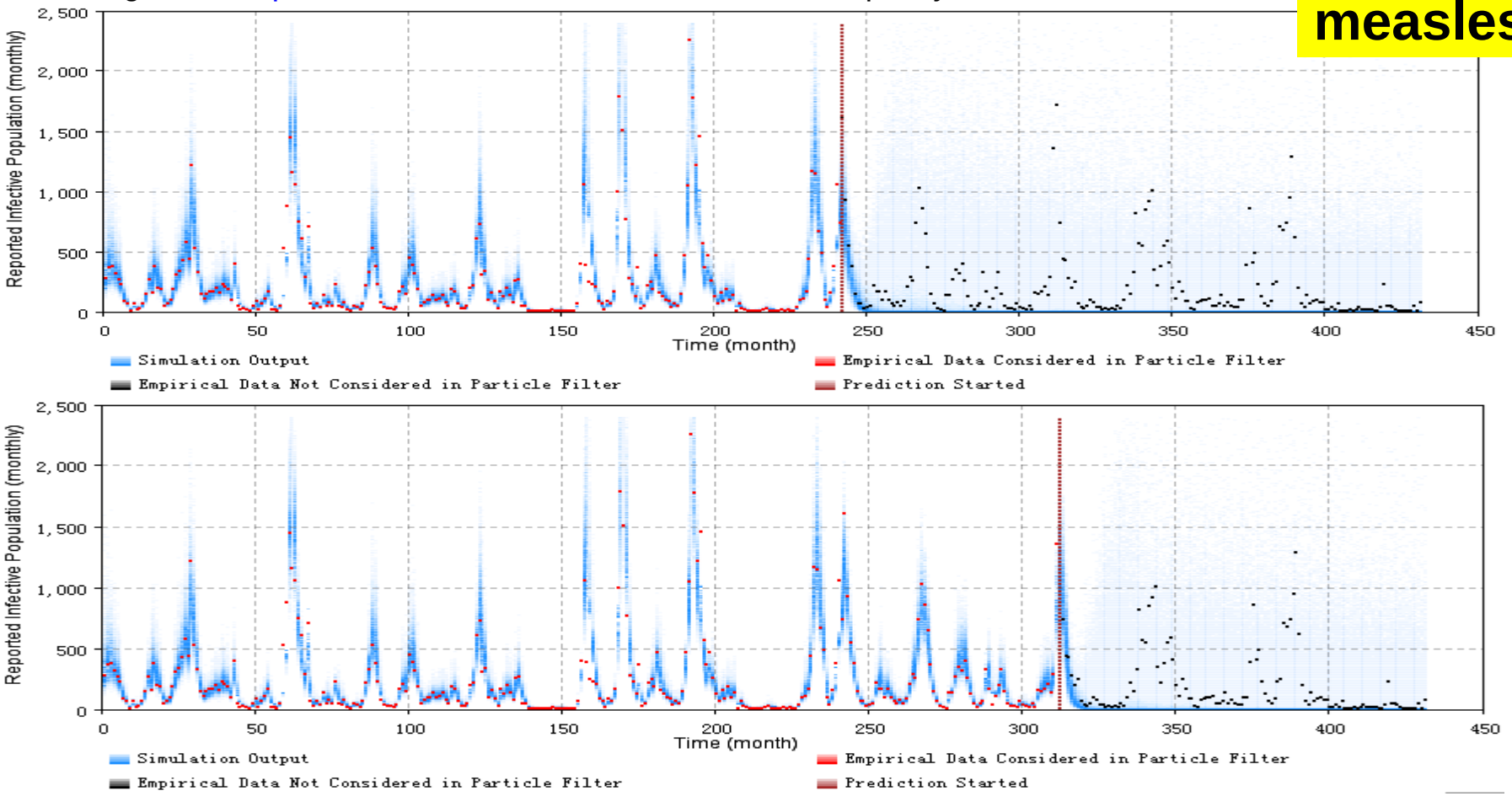
Predicting from the first or second time points of an outbreak of the minimum discrepancy model



Prediction results of the minimal discrepancy model

Predicting from the peak of an outbreak of the minimum discrepancy model

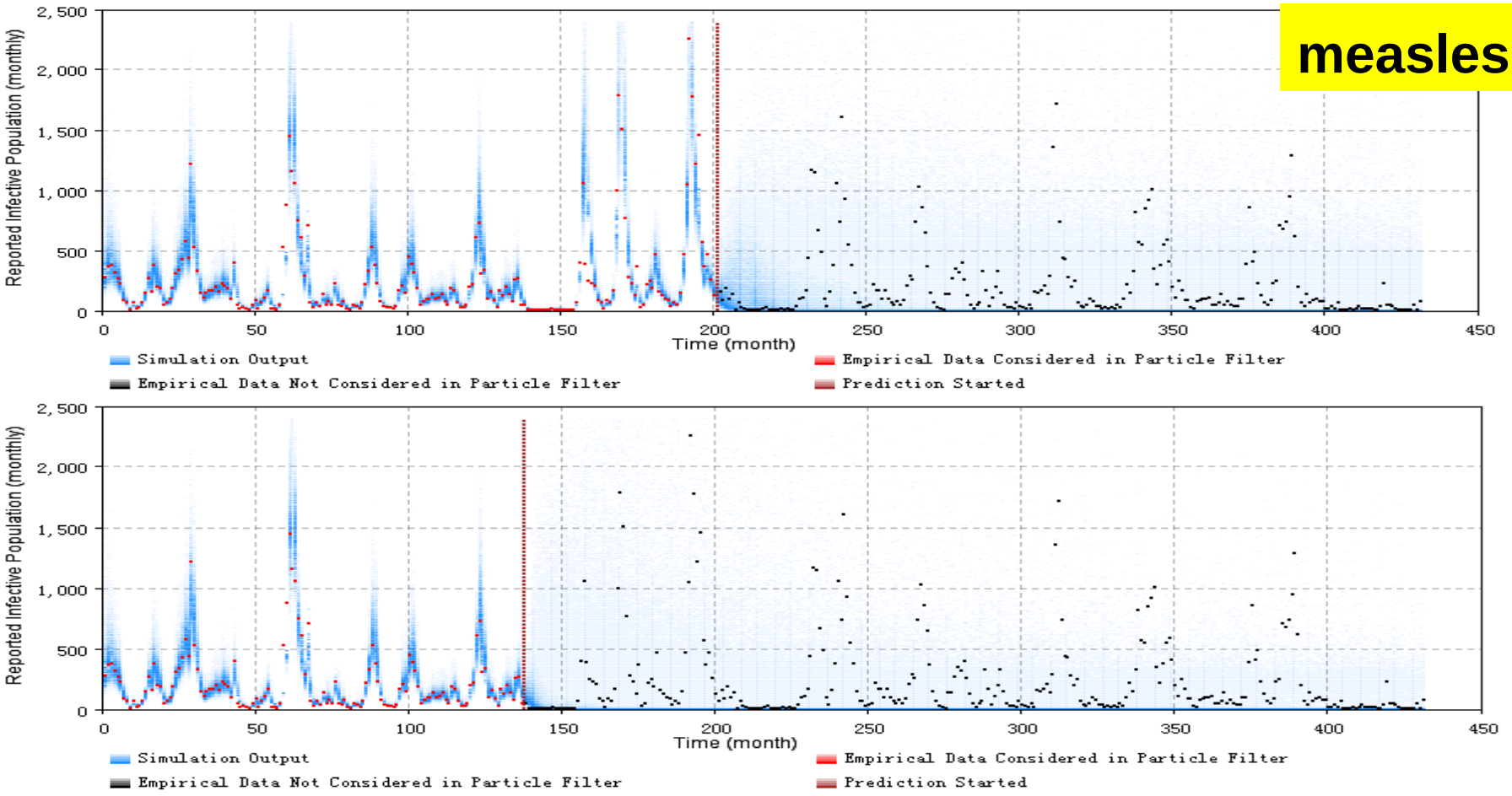
measles



Prediction results of the minimal discrepancy model

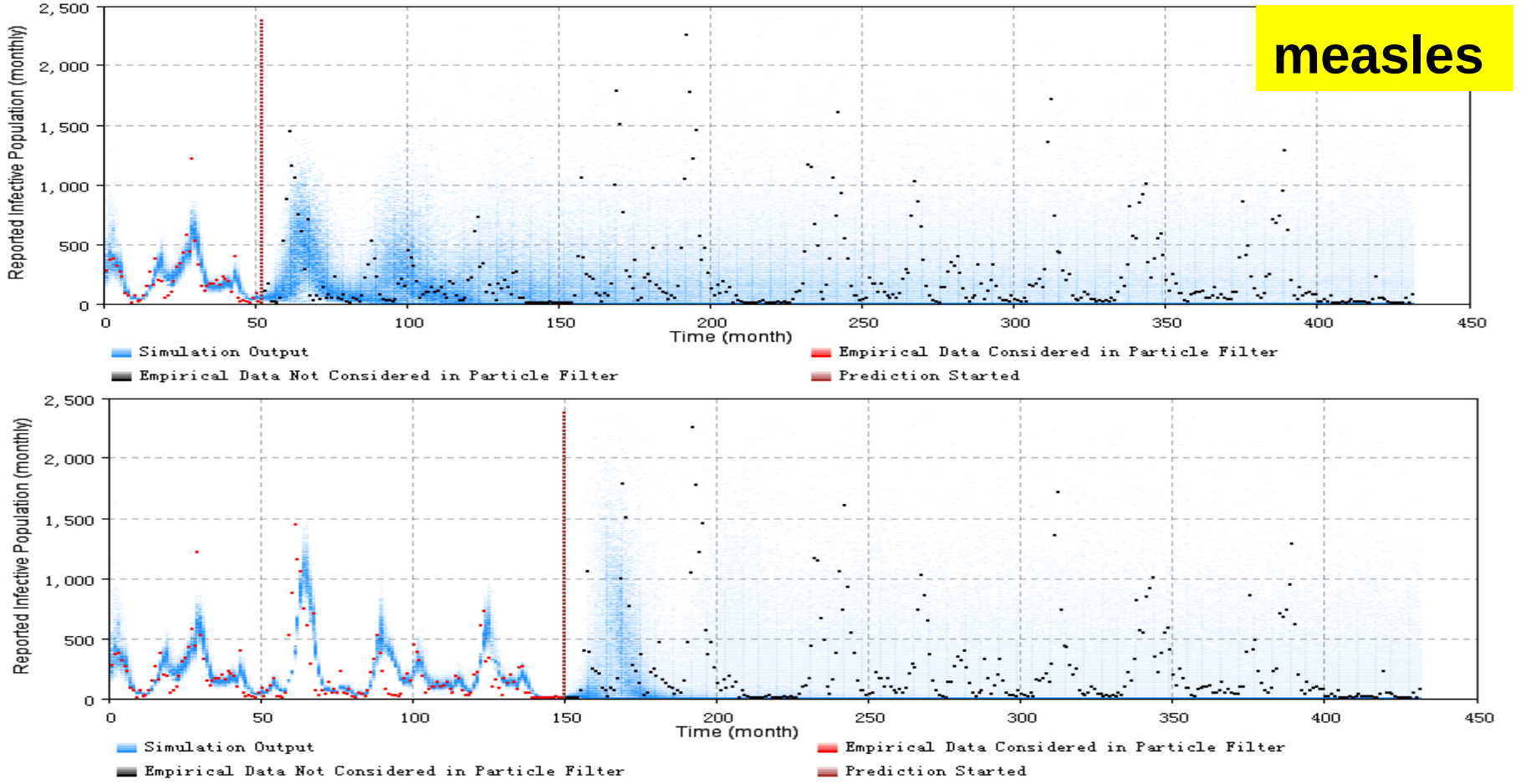
Predicting from the end of an outbreak of the minimum discrepancy model

measles



Prediction results of the minimal discrepancy model

Predicting before the next outbreak of the minimum discrepancy model

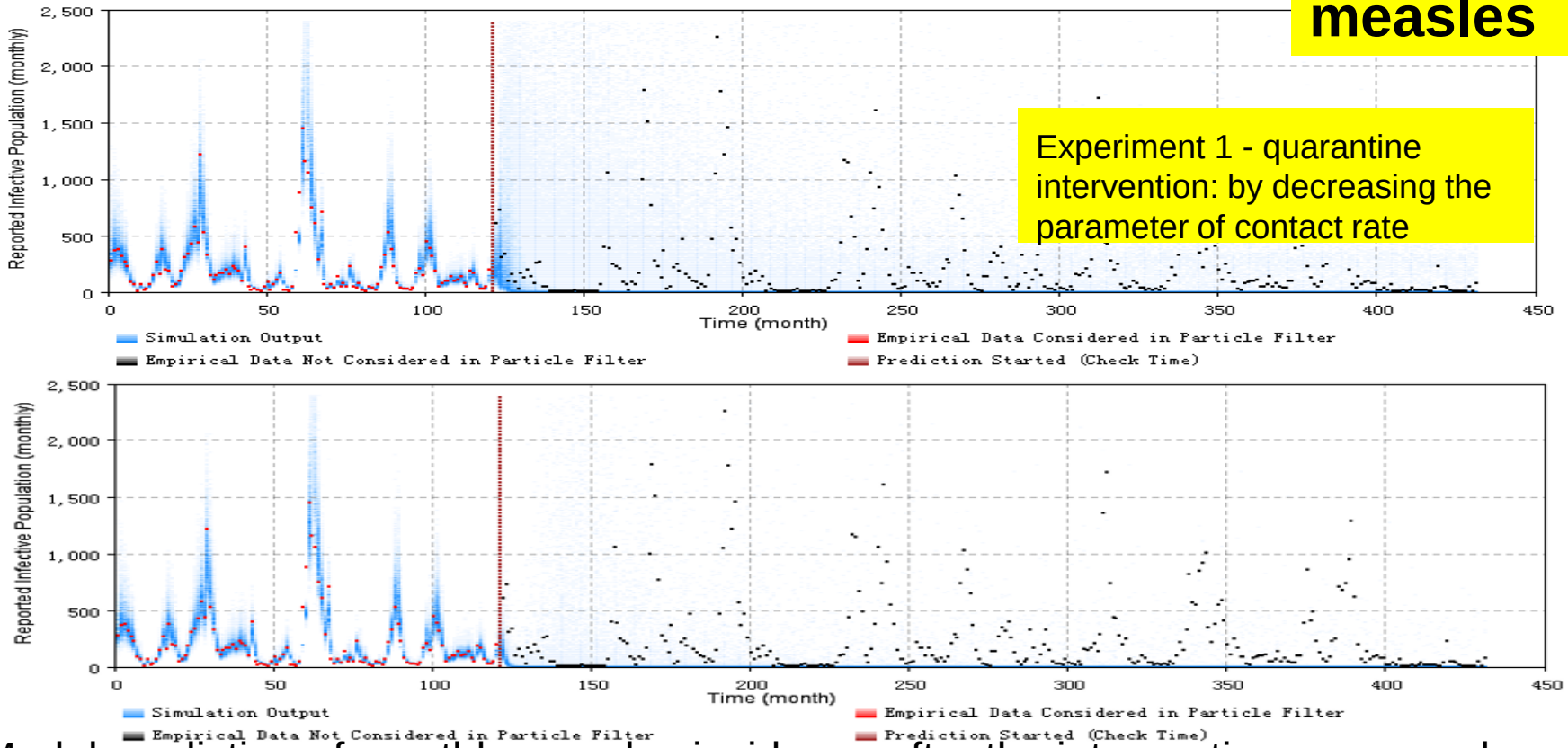


Intervention results of the minimal discrepancy model

simulating an outbreak-response quarantine intervention

measles

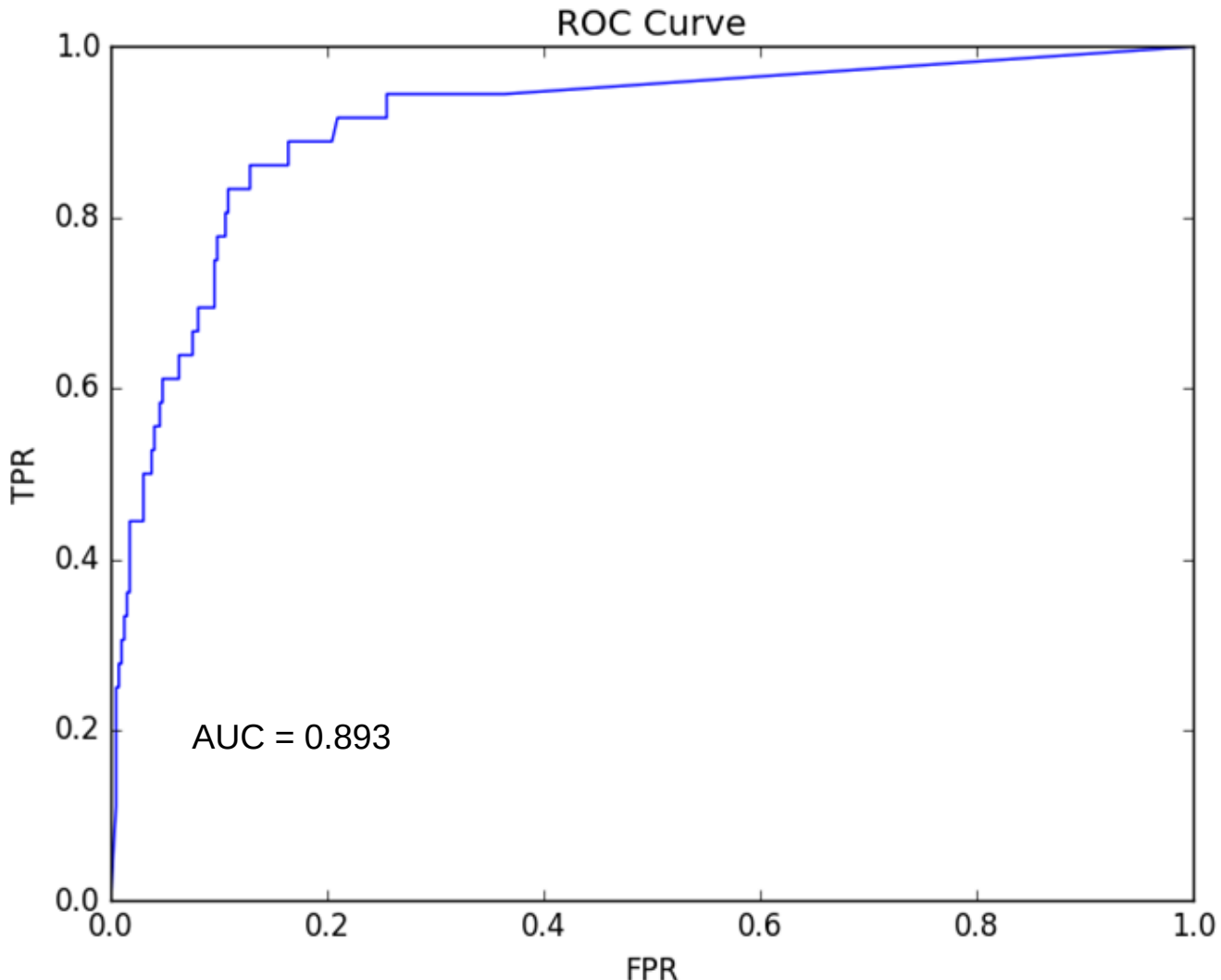
Experiment 1 - quarantine intervention: by decreasing the parameter of contact rate



Model prediction of monthly measles incidence after the intervention expressed as 20-50% (panel A and B respectively) reduction in the contact rate.

Classifying outbreak occurrence with the prediction results of the particle filtering models

measles



Agenda

- ✓Motivations & metaphors
- ✓Particle filtering “how” in a nutshell
- ✓Case study
- Particle filtering: What
- Particle filtering: How
- Balancing model stochastics
- Particle filtering with agent-based models
- Summary

Agenda

- ✓Motivations & metaphors
- ✓Particle filtering “how” in a nutshell
- ✓Case study
- Particle filtering: What
- Particle filtering: How
- Balancing model stochastics
- Particle filtering with agent-based models
- Summary

State Space Modeling

- State space Model $\frac{dx^N}{dt} = g(x^N, \vartheta)$
  - N is the count of state variables
  - X^N represents a vector of length N (generally includes many latent – unobserved or unobservable – elements)
  - ϑ Represents noise in state evolution
- Unit updates: Solution for time t=k is:

$$x_k^N = g_k\left(x_{k-1}^N, \vartheta_{k-1}\right) = \int_{k-1}^{k} g(x^N(t), \vartheta)dt$$

- Where g_k advances the model from *k-1* to *k*

Simplifying Assumptions

- For ease of exposition, we make two assumptions here
  - Making these assumptions allows out exposition to be far simpler
  - These assumptions are for explanation and do not indicate limits to practical particle filter application
- Assumptions:
  - Measurements are made at regular intervals
  - The inter-measurement intervals are of unit length (i.e., measurements are separated by one time unit)
- Thus measurements occur at time k=1, k=2, etc., with measurement k occurring at time k.

Measurement Model

- We generally know a little bit about the underlying state of the system x_k^N at observation time *k* via noisy observations y_k^M
- The measurement model is as follows:

$$y_k^M = h_k(x_k^N, n_k)$$

where

- y_k^M represents a vector of length *M* giving the (noisy & partial) observations at time *k*
- n_k represents the noise affecting that observation
- h_k captures how state x_k^N is abstracted by observa.

Sampling Goal

- At time *k, we* seek to estimate state x_k^N (or, better, trajects $x_{1:k}^N$) based on all of the observed data $y_{1:k}^M$
- The ongoing presence of noise throughout the process prevents naïve application of sampling via e.g., MCMC of the likelihood of observing $y_{1:k}^M$ for a sufficiently broad set of trajectories $x_{1:k}^N$
- To make this feasible, we seek a *recursive* way of estimating (sampling) x_k^N from x_{k-1}^N as a new observation y_k^M occurs
  - This mirrors classic Bayesian updating from a prior to a posterior when a new observation occurs
- This is updated in two stages: **Prediction** & **Update**

What: Two-Phase Recursive Procedure

- **Prediction**: The state x_{k-1}^N (sampled from $p(x_{k-1}^N | y_{1:k-1}^M)$) at time k-1 to is mapped to samples from the state x_{k-}^N (sampled from $p(x_k^N | y_{1:k-1}^M)$) at time *k just* ***prior to*** *the observation* y_k^M.
- **Update**: x_{k-}^N is updated to x_k^N in a way that considers the current observation y_k^M
- **Net effect:** Mapping $p(x_{k-1}^N | y_{1:k-1}^M)$ to $p(x_k^N | y_{1:k}^M)$

What: Two-Phase Recursive Procedure

- **Prediction**: The state x_{k-1}^N (sampled from $p(x_{k-1}^N | y_{1:k-1}^M)$) at time k-1 to is mapped to samples from the state x_{k-}^N (sampled from $p(x_k^N | y_{1:k-1}^M)$) at time *k just **prior to** the observation* y_k^M.
  - For our case (using the Condensation Algorithm), this update does not consider the coming observation y_k^M
- **Update**: x_{k-}^N is updated to x_k^N in a way that considers the current observation y_k^M

Prediction 1

- **Assumption:** p($x_{k-1}^N \mid y_{1:k-1}^M$) is available at time k-1
  - Note that p($x_0^N \mid y_0^M$) = p(x_0^N) is the *prior* for the entire particle filter.
  - We use this ★ to indicate this term in future slides
- Consider that for a binary outcome, B, if we can be confident of the identity:

$$p(A|C) = p(AB|C) + p(A\bar{B}|C)$$

- As formalized in the Chapman–Kolmogorov equation, a variant of this is further true for continuous system. Applying this principle, we can write

$$p\left(x_k^N \middle| y_{1:k-1}^M\right) = \int_{-\infty}^{+\infty} p\left(x_k^N, x_{k-1}^N \middle| y_{1:k-1}^M\right) dx_{k-1}^N$$

This is the analogue of the "+" in the binary situation

Prediction 2

- Consider the identity just shown:

$$p\left(x_k^N \middle| y_{1:k-1}^M\right) = \int_{-\infty}^{+\infty} p\left(x_k^N, x_{k-1}^N \middle| y_{1:k-1}^M\right) dx_{k-1}^N$$

- By the probabilistic chain rule, in general

$$p(AB|C) = \frac{p(ABC)}{P(C)} = \frac{p(A|BC)P(BC)}{P(C)} = p(A|BC)P(B|C)$$

- Thus we can rewrite the above as:

$$\int_{-\infty}^{+\infty} \mathrm{p}\left(x_k^N \middle| x_{k-1}^N, y_{1:k-1}^M\right) p\left(x_{k-1}^N \middle| y_{1:k-1}^M\right) dx_{k-1}^N$$

We call that in planning the recursive procedure,
We assumed (in the previous slide) that sampling from this term is possible at time k-1!

Prediction 3

- For a first-order Markov process such as this, the impact of past observations $y_{1:k-1}^{M}$ earlier than time *k-1* **is imparted purely through their influence on state x_{k-1}^{N}.** Thus

$$p\left(x_k^N \middle| x_{k-1}^N, y_{1:k-1}^M\right) = p\left(x_k^N \middle| x_{k-1}^N\right)$$

Thus the previous formula

$$\int_{-\infty}^{+\infty} \mathrm{p}\left(x_k^N \middle| x_{k-1}^N, y_{1:k-1}^M\right) p\left(x_{k-1}^N \middle| y_{1:k-1}^M\right) dx_{k-1}^N$$

Can be rewritten as

$$p\left(x_k^N \middle| y_{1:k-1}^M\right) = \int_{-\infty}^{+\infty} p\left(x_k^N \middle| x_{k-1}^N\right) p\left(x_{k-1}^N \middle| y_{1:k-1}^M\right) dx_{k-1}^N$$

Given x_{k-1}^N, we sample from this by just simulating forward from time *k*-1 to time *k*!

These are just the samples available at time *k*-1 (once having processed the observation y_{k-1}^M)

Dynamics at Observations

- At observations, the model estimates of state are "corrected" by empirical data
- This transitions from the "prior" to the "posterior"
  - Prior: Estimate of model state immediately before considering the latest empirical data (NB: this was produced by running model)
  - Posterior: Estimate of model state immediately after considering the new empirical data

Likelihood Function Explored

- Likelihood functions $p(y_t^M | x_t^N)$ give the likelihood of the empirical datum, given the model state
- Common distributions
  - Binomial
  - Negative
  - Poisson
  - Normal
  - Lognormal

What: Two-Phase Recursive Procedure

- **Prediction**: The state x_{k-1}^N (sampled from $p(x_{k-1}^N | y_{1:k-1}^M)$) at time k-1 to is mapped to samples from the state x_{k-}^N (sampled from $p(x_k^N | y_{1:k-1}^M)$) at time *k just **prior to** the observation* y_k^M.
  - For our case (using the Condensation Algorithm), this update does not consider the coming observation y_k^M
- **Update**: x_{k-}^N is updated to x_k^N in a way that considers the current observation y_k^M

Update 1

- The prediction phase provides the capacity to sample from $\mathrm{p}\left(x_k^N \middle| y_{1:k-1}^M\right)$
  - i.e., given y_{k-1}^M the prediction phase updates samples from $\mathrm{p}(x_{k-1}^N | y_{1:k-1}^M)$ to samples from $\mathrm{p}\left(x_k^N \middle| y_{1:k-1}^M\right)$
  - In future slides, we indicate this term $\mathrm{p}\left(x_k^N \middle| y_{1:k-1}^M\right)$ using
- The update phase maps samples from $\mathrm{p}\left(x_k^N \middle| y_{1:k-1}^M\right)$ to samples from $\mathrm{p}\left(x_k^N \middle| y_{1:k}^M\right)$
  - This is just updated from time *k*-1 to time *k*, which we indicate using the symbol
  - Here, we have just taken into account information from the observable y_k^M
- It turns out that sampling from $\mathrm{p}\left(x_k^N \middle| y_{1:k}^M\right)$ can be readily achieved by exploiting the fact that

$$p\left(x_k^N \middle| y_{1:k}^M\right) \propto p(y_k^M | x_k^N) p(x_k^N | y_{1:k-1}^M)$$

Sampling from the full trajectory

- Recall from "Update 10" that

$$p\left(x_k^N \middle| y_{1:k}^M\right) \propto p(y_k^M | x_k^N) p(x_k^N | y_{1:k-1}^M)$$

By extension, we have

$$p\left(x_{0:k}^N \middle| y_{1:k}^M\right) \propto p(y_k^M | x_{0:k}^N) p(x_{0:k}^N | y_{1:k-1}^M)$$

Recognizing that $x_{0:k}^N$ is composed of pieces, we rewrite this as

$$p(y_k^M | x_{0:k}^N) p(x_k^N, x_{0:k-1}^N | y_{1:k-1}^M)$$

By the probabilistic chain rule, in general

$$p(AB|C) = \frac{p(ABC)}{P(C)} = \frac{p(A|BC)P(BC)}{P(C)} = p(A|BC)P(B|C)$$

Thus we can rewrite the above as

$$p\left(x_{0:k}^N \middle| y_{1:k}^M\right)$$
$$\propto p\left(y_k^M \middle| x_k^N, x_{0:k-1}^N\right) p\left(x_k^N \middle| x_{0:k-1}^N, y_{1:k-1}^M\right) p(x_{0:k-1}^N | y_{1:k-1}^M)$$

Full Trajectory Distribution

- From the previous slide, we have
$$p\left(x_{0:k}^{N}\middle|y_{1:k}^{M}\right) \propto p\left(y_{k}^{M}\middle|x_{k}^{N},x_{0:k-1}^{N}\right)p\left(x_{k}^{N}\middle|x_{0:k-1}^{N},y_{1:k-1}^{M}\right)p\left(x_{0:k-1}^{N}\middle|y_{1:k-1}^{M}\right)$$
- because we assume a first-order Markov process, we recognize that two terms above can be simplified
$$p\left(y_{k}^{M}\middle|x_{k}^{N},x_{0:k-1}^{N}\right) = p\left(y_{k}^{M}\middle|x_{k}^{N}\right)$$
$$p\left(x_{k}^{N}\middle|x_{0:k-1}^{N},y_{1:k-1}^{M}\right) = p\left(x_{k}^{N}\middle|x_{k-1}^{N}\right)$$
- Substituting these in above, we have
$$p\left(x_{0:k}^{N}\middle|y_{1:k}^{M}\right) \propto p\left(y_{k}^{M}\middle|x_{k}^{N}\right)p\left(x_{k}^{N}\middle|x_{k-1}^{N}\right)p\left(x_{0:k-1}^{N}\middle|y_{1:k-1}^{M}\right)$$

Agenda

- ✓Motivations & metaphors
- ✓Particle filtering "how" in a nutshell
- ✓Case study
- ✓Particle filtering: What
- Particle filtering: How
- Balancing model stochastics
- Particle filtering with agent-based models
- Summary
- Supplemental material: Detailed documentation & derivations

Shifting Gears: From "What" to "How"

- The previous section indicated the "target distribution" from which have to sample
- This is a distribution $p\left(x_{0:k}^{N} \middle| y_{1:k}^{M}\right)$ over trajectories $x_{0:k}^{N}$ of latent states of the system, given observations $y_{1:k}^{M}$
- This section examines the mathematics underlying the scheme for *how we actually go about sampling from that distribution*

Central Role of Importance Sampling

- Particle Filtering uses another means of sampling – sequential importance sampling
- Here, we forsake the goal of sampling directly from the ideal ("target") distribution via MCMC
- Rather, we use the importance sampling technique of
  - Sampling readily from a different "proposal distribution"
  - Using weighting of the samples from the proposal distribution to characterize their relative representation in the target relative to the proposal distribution
  - This weight allows us to sample more frequently from those samples that are well represented in the target distribution, and less frequently from others
- The coming slides present the theory behind importance sampling & sequential importance sampling

Importance Sampling 1

- Suppose that we want to draw samples x from a “target” distribution p(x) that
  - Is difficult to sample from directly
  - Given an x, has a value that can be readily computed
- Suppose that we have another “proposal” distribution q(x) from which we can readily sample
  - Common example: Uniform distribution
  - Ideally this would be something like p(x)

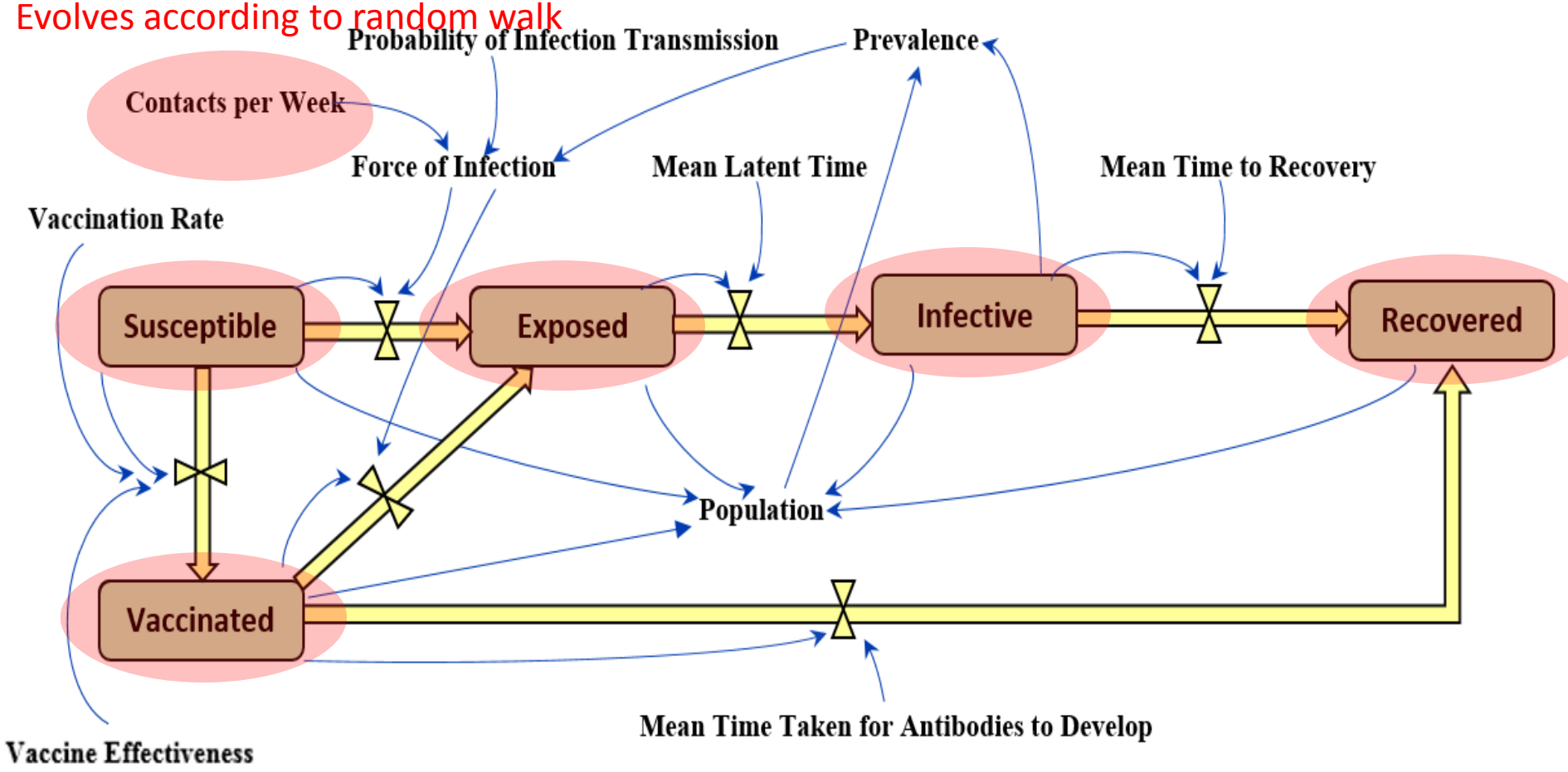
Importance Sampling 2

- We can readily sample *N* values x from p(x) by a 4 part procedure:
  - Part 1: Create a set S of N sample values x_i ($1 \le i \le N$) from q(x)
    - e.g., draw each x_i from a uniform distribution between 0 and 1
  - Part 2: Label each drawn value x_i with a "weight" $W_i = p(x_i)/q(x_i)$
    - This "weight" expresses how much more common x_i is within the target distribution p(x), when compared to the distribution from which it was drawn, q(x)
  - Part 3: Normalize weights, labeling each x_i with weight $w_i = \frac{W_i}{\sum_{j=1}^{N} W_j}$
  - Part 4: Draw N samples from S, where the probability of drawing sample i each is proportional to w_i
    - For each such sample, this is readily performed by
      - Drawing a value *u* from [0,1]
      - Going through each $1 \le i \le N$ accumulating the value of w_i until the smallest i where $\sum_{j=1}^{i} W_j \ge u$

How: Particles in Particle Filtering

- Each importance-weighted sample is represented by a "particle"
  - At any one time, the particles represent a sampling from the state of that model at that time
  - We can compute statistics on ***sampled*** particles to approximate applying such statistics to the full distribution
- Each particle is associated with a
  - Copy of model state (anything over which seeking distr.); here, values of state variables (stocks – S,E,I,R & contact rate)
  - Normalized weight (like weight in ***importance sampling***)
- "Survival of the fittest": Particles reproduce and survive or perish according to their quality of match to the empirical observations (weight)
  - ***Learning***: Trajectories more consistent with the data survive
- There is typically a fixed number of particles retained through the simulation

Elements of Model State Associated with Particles



Hint: Think of the model as having a vertical ("Z") direction, with each state variable representing a "stack" of state variables, one for each particle

Back

How: Particle Dynamics: Two Components

- Recall: Each particle has its own full copy of model state (anything that could differ b/t realizations)
- **Prediction:** Between observations
  - Particles evolve according to standard [stochastic] model dynamics (just run the model on each particle's state)
  - Particle weights remain invariant
  - There is no filtering out of particles
- **Update:** At observation point
  - Particle weights are updated to reflect likelihood of observing the empirical data, given the particle state
  - If too much disparity in weights, particles are resampled according to their weights; tendency:
    - Particles w/ high weights reproduce; w/low weights disappear

Overall Algorithm

At time $t \geq 2$, perform a recursive update as follows

(1) Advance the sampled state by sampling $X_t^{(i)} \sim q_t(x_t | y_t, X_{1:t-1}^{(i)})$ and set $X_{1:t}^{(i)} = (X_{1:t-1}^{(i)}, X_t^{(i)})$;

(2) Update the weights to reflect the probabilistic and state update models

$$w_t^{(i)} = w_{t-1}^{(i)} \frac{p(X_t^{(i)} | X_{t-1}^{(i)}) g(y_t | X_t^{(i)})}{p(X_t^{(i)} | X_{t-1}^{(i)})}$$

g is the likelihood function: Gives likelihood of the empirical datum y_t, given the model state X_t

Normalize the weights $W_t^{(i)} = \frac{w_t^{(i)}}{\sum_{i=1}^{N} w_t^{(i)}}$.

For any time t, if the effective sample size is too small (i.e., the variance of the weights is too high, $\frac{1}{\sum_{i=1}^{N} (W_t^{(i)})^2} < K$), resample $X_t^{(i)}$ and set $W_t^{(i)} = \frac{1}{N}$. Here K is a threshold value for the variation of the weights.

How: **Prediction**
(Dynamics Between Observations)

- Recall: Each particle has its own full copy of model state (anything that could differ b/t realizations)
- Each particle runs the model forward until the next observation
  - Originally identical particles diverge because of
    - Model stochastics
    - Distribution over some parameters
  - For our model, the particle evolution is governed by a stochastic differential equation
- Weights remain unchanged

Convenient Choice in Proposal Distribution

- *For convenience, we assume that*

$$q\left(x_{1:k}^{N}\middle|y_{1:k}^{M}\right)=q\left(x_{k}^{N}\middle|x_{1:k-1}^{N},y_{1:k}^{M}\right)q\left(x_{1:k-1}^{N}\middle|y_{1:k-1}^{M}\right)$$

- These terms represent
  - $q\left(x_{1:k-1}^{N}\middle|y_{1:k-1}^{M}\right)$ is just the value for the proposal distribution **at the time of the previous observations**
  - $x_{k}^{N(i)}\sim q(x_{k}^{N}|x_{1:k-1}^{N},y_{1:k}^{M})$: the probability (density) of the updated state taking into account the new measurement measurements at time *k*

Update Phase: Dynamics at Observations

- N: Count of state variables (stocks)
- M: Count of observations
- Because we can't easily draw a sample from the posterior distribution over trajectories $p\left(x_k^N \middle| y_{1:k}^M, x_{k-1}^N\right)$, we use importance sampling
  - We actually capture the posterior by updating particle weights, and possibly resampling (see later)
  - This involves choosing a proposal distribution $q\left(x_k^N \middle| y_t, x_{k-1}^N\right)$
- Please note that we will often go beyond the above and sample from the full trajectories $p\left(x_{0:k}^N \middle| y_{1:k}^M\right)$ rather than simply from the value at time *k*

Sequential Importance Sampling for Our Case

- Importance-weighted samples are maintained over time
  - Such samples are termed “particles”
- Successive observations at integer times are made, with each such observation updating the underlying distribution $\mathrm{p}\left(x_{0:k}^{N(i)} \middle| y_{1:k}^{M}\right)$
  - Direct sampling from this distribution is not generally possible
- To enable successively sample from $\mathrm{p}\left(x_{0:k}^{N(i)} \middle| y_{1:k}^{M}\right)$ as each observations *k* arrives, we draw instead from a proposal distribution *q(x)* & successively update samples $x_{1:k-1}^{N(i)}$ & weights $w_{k-1}^{(i)}$ to reflect the observation, yielding $x_{1:k}^{N(i)}$ and $w_{k}^{(i)}$

Importance Sampling to Draw From

- Target distribution is $p\left(x_{0:k-1}^{N(i)} \middle| y_{1:k-1}^{M}\right)$
- $x_{1:k-1}^{N(i)}$ is drawn from proposal distribution $q\left(x_{0:k-1}^{N(i)} \middle| y_{1:k-1}^{M}\right)$
- Here we are seeking to choose and maintain a *proposal distribution* that can be readily sampled from over time
  - Per importance sampling principles, these samples are then weighted as

$$w_{k-1}^{(i)} = \frac{p\left(x_{0:k-1}^{N(i)} \middle| y_{1:k-1}^{M}\right)}{q\left(x_{0:k-1}^{N(i)} \middle| y_{1:k-1}^{M}\right)}$$

Recursive Weight Updates Redux

- We have $w_{k-1}^{(i)} = \frac{p\left(x_{0:k-1}^{N(i)} \middle| y_{1:k-1}^{M}\right)}{q\left(x_{0:k-1}^{N(i)} \middle| y_{1:k-1}^{M}\right)}$
- As demonstrated in the supplemental slides, we can then take advantage of several factors
  - The assumed form of the proposal distribution
  - The formula for the probability of the trajectories $p\left(x_{0:k}^{N} \middle| y_{1:k}^{M}\right)$
  - Cancellation
  - The fact that the process g_k is first-order Markovian (and thus the current state the past state)
- The formula for the weight can then be formulated recursively as:

$$w_k^{(i)} \propto w_{k-1}^{(i)} \frac{p\left(y_k^M \middle| x_k^{N(i)}\right) p\left(x_k^{N(i)} | x_{k-1}^{N(i)}\right)}{q\left(x_k^{N(i)} \middle| x_{k-1}^{N(i)}, y_k^M\right)}$$

Recursive Weight Updates Redux

- Supplemental slides demonstrate that:

$$w_k^{(i)} = \frac{p\left(y_k^M \middle| x_k^N\right) p\left(x_k^N \middle| x_{k-1}^N\right)}{q\left(x_k^N \middle| x_{1:k-1}^N, y_{1:k}^M\right)} w_{k-1}^{(i)}$$

- Because the process g_k is first-order Markovian, we can further simplify the denominator by recognizing that all impact of past state is captured in the previous state $x_{k-1}^{N(i)}$, and by assuming that all observations prior to k have already between reflected in the distribution, thus:

$$q\left(x_k^{N(i)} \middle| x_{1:k-1}^{N(i)}, y_{1:k}^M\right) = q\left(x_k^{N(i)} \middle| x_{k-1}^{N(i)}, y_k^M\right)$$

and (rearranging slightly) we have

$$w_k^{(i)} \propto w_{k-1}^{(i)} \frac{p\left(y_k^M \middle| x_k^{N(i)}\right) p\left(x_k^{N(i)} | x_{k-1}^{N(i)}\right)}{q\left(x_k^{N(i)} \middle| x_{k-1}^{N(i)}, y_k^M\right)}$$

Naïve Algorithm

Initialization (k=0)

- Sample $X_0^{N(i)}$ from $q_0(x_0^N | y_0^M)$
- *For each particle i*
  - $w_k^{(i)} = \frac{p(X_0^{N(i)}) p(y_0^M | X_0^{N(i)})}{q(X_0^{N(i)} | y_0^M)}$
- For each particle i
  - $W_0^{(i)} = \frac{w_0^{(i)}}{\sum_{i=1}^{N_S} w_0^{(i)}}$

Ongoing Observations (k>0)

- For each particle i
  - Advance state via sampling $X_k^{N(i)} \sim q(x_k^N | y_k, X_{0:k-1}^{N(i)})$
  - Supplement trajectory $X_{0:k}^{N(i)} = (X_{0:k-1}^{N(i)}, X_k^{N(i)})$
  - Update weight

$$w_k^{(i)} = w_{k-1}^{(i)} \frac{p\left(y_k^M \middle| x_k^{N(i)}\right) p\left(x_k^{N(i)} | x_{k-1}^{N(i)}\right)}{q\left(x_k^{N(i)} \middle| x_{k-1}^{N(i)}, y_k^M\right)}$$

- For each particle i

$$W_k^{(i)} = \frac{w_k^{(i)}}{\sum_{i=1}^{N_s} w_k^{(i)}}$$

Choice of Proposal Distribution

- The algorithms above leave open to the implementer the choice of a proposal distribution

$$q(x_k^N | y_k, X_{0:k-1}^{N(i)})$$

- The choice of proposal distribution can have a sizeable impact on
  - The practical performance of the particle filtering
  - The complexity of the implementation
- Many practitioners make use of the “condensation algorithm” of Isard & Blake (1998), which employs a particularly simple proposal distribution

Condensation Algorithm

- Goes by additional names in the computational statistics literature
- Involves just using the prior distribution until the next point of observation, and then updating at that time using the likelihood
- In operational terms, this means that we simply
  - Run the model forward from the last data point (this samples from "the prior") until the next observation point
  - At the time of the observation, multiply the weight of a particle by the value of the likelihood of observing the datapoint given the model state hypothesized by that particle
- It is not yet clear to the instructor if there is an effective and attractive alternative for a non-linear model
  - Any such alternative would certainly be considerably more involved

The "*condensation algorithm*" *(Isard & Blake 1998)*

- Here, one choses $q\left(x_k^{N(i)}\middle|x_{k-1}^{N(i)}, y_k^M\right) = p(x_k^{N(i)}|x_{k-1}^{N(i)})$
  - In other words, the **proposal** distribution simply uses the simulation induced probability distribution and ignores the data
- Recall that, in general, for particle *i*
  - $w_k^{(i)} = w_{k-1}^{(i)} \frac{p\left(y_k^M\middle|x_k^{N(i)}\right)p(x_k^{N(i)}|x_{k-1}^{N(i)})}{q\left(x_k^{N(i)}\middle|x_{k-1}^{N(i)}, y_k^M\right)}$

We are just multiplying the old weight by the likelihood to get the new weight!

- For $q\left(x_k^{N(i)}\middle|x_{k-1}^{N(i)}, y_k^M\right) = p(x_k^{N(i)}|x_{k-1}^{N(i)})$:

$$w_k^{(i)} = w_{k-1}^{(i)} \frac{p\left(y_k^M\middle|x_k^{N(i)}\right)p(x_k^{N(i)}|x_{k-1}^{N(i)})}{p(x_k^{N(i)}|x_{k-1}^{N(i)})} = w_{k-1}^{(i)} p\left(y_k^M\middle|x_k^{N(i)}\right)$$

- Recall further that $p\left(y_k^M\middle|x_k^{N(i)}\right)$ is the **likelihood** of observing the empirical data at that time (y_t) given the particle state $x_k^{N(i)}$ at that time

Recall:
Central Choice: Likelihood Function(s)

- The likelihood function expresses the likelihood of observing the empirical data in light of particle-posited model state
  - In the condensation algorithm, this is applied (only) when we receive a new observation
  - Given a view of particle filtering as a “survival of the fittest” our likelihood function dictates what “fit” means (dictates “how good a match” there is to observed data)
- The choice of one or more likelihood functions is one of the most important elements in design of an effective particle filter model

Forms of Likelihood Functions Explored

- Binomial
  - Not recommended because gives 0 likelihood if all particles posit values for the count of trials ("coin flips") that are less than the observed value
- Negative Binomial

$$y_t | x^s_t \equiv y_t | i_t \sim \text{NegativeBinomial}(i_t, r)$$

This $p(y_t | i_t)$ is non-zero for all $i_t > 0$

- Normal
  - Here, we are matching against values that could in general be negative

Dealing with Multiple Types of Observations

- To deal with multiple types of observations, we require a multivariate likelihood function
- Common simplification: Take product of observation-specific likelihood functions

Incremental (Recursive) Nature of Updates & Streaming

- Particle filtering with condensation algorithm involves just incremental ("recursive") updates to the weights over time
  - When a new observation comes in, we just take the old weight and multiply by the value of the likelihood function applied to the
    - Current state
    - New observation
- Do *not* have handle all observations jointly at a time
- This incremental nature of the updates makes the process very well suited to *streaming* solutions that handle each new observation vector as it arrives

Resampling Step

- When there is too large a diversity of particle weights following the updating of weights, we *perform a (weighted) resampling* from the particles
  - We draw a new set of particles from the set of particles (reflecting the updated weights), where the chance of selecting a given particle is proportional to is weight
  - A given particle may be disappear or be duplicated many times
    - NB: If it is duplicated several times, note that the resulting particles have a complete copy of the state of the original particle (such that this state can then evolve independently)
  - The resampled particles are assigned a weight of 1

Practical Problem: Reduced Effective Sample Size

- Performed naively, the algorithm above can lead to a situation where
  - Most particles have very low weight
  - Only a few particles have significant weight
- This situation gives a low "effective sample size" in that with sequential importance sampling, the high weight particles will be overwhelmingly overrepresented
- We can recognize this situation by monitoring the variance in weights, using the second moment of the weights

$$S_{eff} = \frac{1}{\sum_{i=1}^{N} (w_k^{(i)})^2}$$

- If $S_{eff} < S_T$, we view the samples as suffering too low a sample size, and perform resampling

Resampling Step

- *Suppose we have particles* $X_k^{N(i)}$ (call these $X_{k-}^{N(i)}$) whose weight $W_{k-}^{(i)}$ has just been updated by an observation;
- Let $S_{eff} = \frac{1}{\sum_{i=1}^{N}(W_k^{(i)})^2}$
- If $S_{eff} < S_T$, then $\forall i, 1 \leq i \leq N$
  - $X_{k+}^{N(i)} \sim X_{k-}^{N(multinomial(W_{k-}^{(1..N)}))}$
  - $W_{k+}^{(i)} = 1/N$
- Please note that $multinomial(W_{k-}^{(1..N)})$ represents a sample from the multinomial distribution, taking as arguments N parameters representing the probabilities of returning each value (here $W_{k-}^{(1..N)}$) and returning an index of the chosen value

Algorithm Maintaining Sample Size

Initialization (k=0)

- Sample $X_0^{N(i)}$ from $q_0(x_0^N | y_0^M)$
- *For each particle i*
  - $w_k^{(i)} = \frac{p(X_0^{N(i)}) p(y_0^M | X_0^{N(i)})}{q(X_0^{N(i)} | y_0^M)}$
- For each particle i
  - $W_0^{(i)} = \frac{w_0^{(i)}}{\sum_{i=1}^{N_S} w_0^{(i)}}$

Ongoing Observations (k>0)

- For each particle i
  - Advance state via sampling $X_k^{N(i)} \sim q(x_k^N | y_k, X_{0:k-1}^{N(i)})$
  - Supplement trajectory $X_{0:k}^{N(i)} = (X_{0:k-1}^{N(i)}, X_k^{N(i)})$
  - Update weight

$$w_k^{(i)} = w_{k-1}^{(i)} \frac{p\left(y_k^M \middle| x_k^{N(i)}\right) p\left(x_k^{N(i)} \middle| x_{k-1}^{N(i)}\right)}{q\left(x_k^{N(i)} \middle| x_{k-1}^{N(i)}, y_k^M\right)}$$

- For each particle i

$$W_k^{(i)} = \frac{w_k^{(i)}}{\sum_{i=1}^{N_S} w_k^{(i)}}$$

- Let $S_{eff} = \frac{1}{\sum_{i=1}^{N_S} (W_k^{(i)})^2}$
- If $S_{eff} < S_T$, resample $X_k^{N(i)}$

Resampling

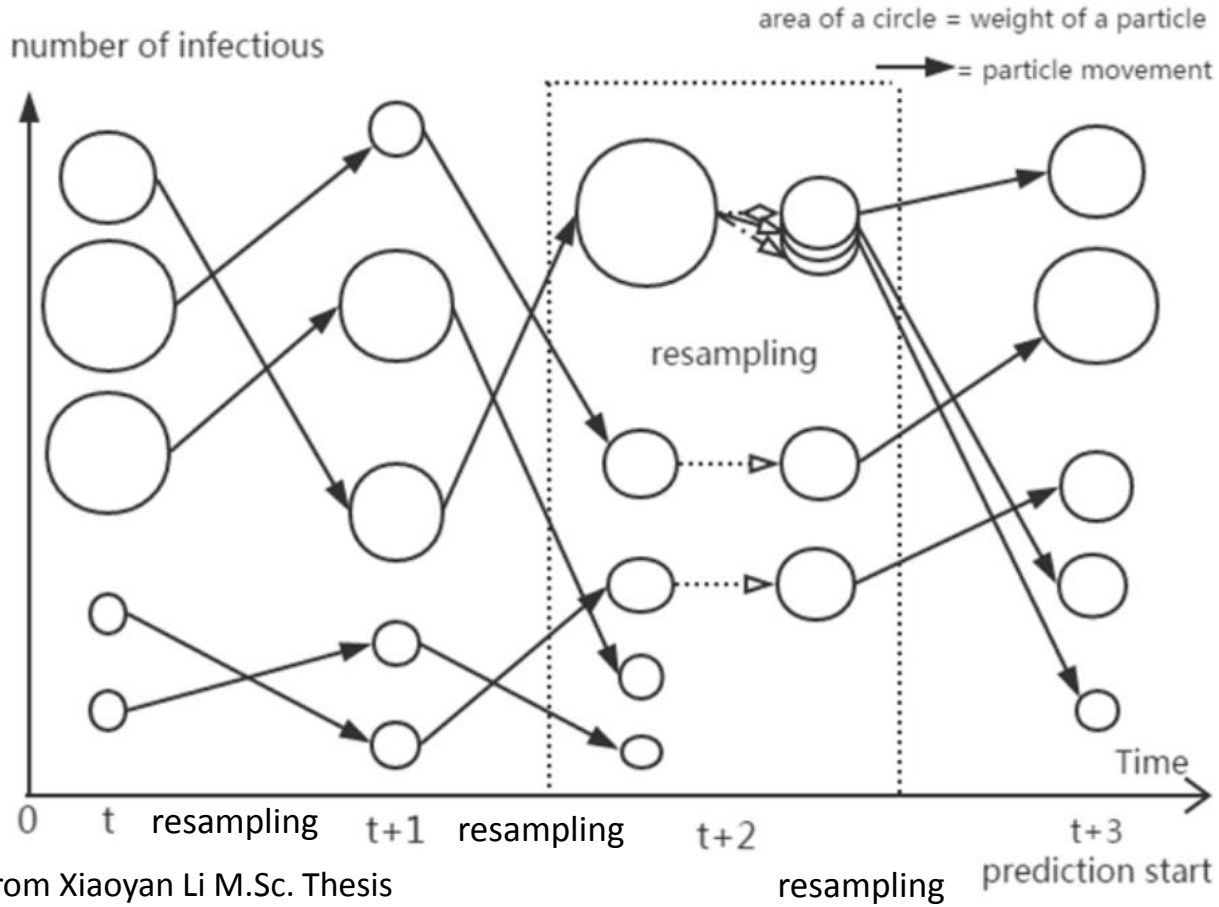


Image taken from Xiaoyan Li M.Sc. Thesis

Computing Statistics over Model Quantities

- If we wish to compute statistics over the model output, we must do so over *samples* from the weighted particles
  - That is, we draw the samples with replacement from the values of the particles, where the probability of drawing a particle is proportional to its weight
- Because particles are samples from the proposal distribution (rather than from the target distribution), we should not be computing statistics on the particles themselves, but on samples from them

Agenda

- ✓Motivations & metaphors
- ✓Particle filtering "how" in a nutshell
- ✓Particle filtering: What
- ✓Particle filtering: How
- Balancing model stochastics
- Particle filtering with agent-based models
- Summary
- Supplemental material: Detailed documentation & derivations

Uncertainty: A Key Balance

- Sometimes such stochastics characterize particular known stochastic processes (e.g., evolution of reporting or contact rates)
- Sometimes stochastics seem to play mostly an instrumental role in achieving model "humility" (breadth of possibilities across particles) without characterizing specific known stochastic phenomena
  - We just use to avoid model overconfidence
- We are currently investigating alternative means of adding in variety & stochastics to the model

Stochastics: A Key Balance

- We need some stochastics in the model, or else all particles cloned during resampling will evolve identically, with no divergence
- **Avoid overconfidence**: We require enough model stochastics to allow the model to have a requisite variety to match a wide variety of different data
  - Too narrow a distribution will lead to "overconfidence" in model predictions – will not be as open to correction by "surprising" data
- **Avoid underconfidence**: We don't want the model to have such pronounced stochastics that, absent data, it quickly becomes hopelessly uncertain

Two common means of introducing stochastics

- Via having parameter values represented as stocks, and allowing them or transforms to evolve via random walks over time (e.g., reporting rate, contact rate)
- Stochastics in processes (e.g., a distribution of count of people infected, around the mean)

Dangers

- Too few particles
  - e.g., if have large number of uncertain input parameters
- Particle impoverishment
  - Particles become too small in number
- Condensation algorithm is too naïve
  - e.g., if new data is received very frequently compared to how quickly the growth in model-related uncertainty

Agenda

- ✓Motivations & metaphors
- ✓Particle filtering “how” in a nutshell
- ✓Particle filtering: What
- ✓Particle filtering: How
- ✓Balancing model stochastics
- Particle filtering with agent-based models
- Summary
- Supplemental material: Detailed documentation & derivations

Particle Filtering with ABMs

- Guidelines for effective particle filtering with ABMs have yet to be elucidated
- Given high nominal (& likely moderately high intrinsic) dimensionality of state space, non-sparse coverage requires high # of particles
- Exceptionally weighty computational resource demand
  - High dimensionality $\Rightarrow$ High number of particles
  - Per-ensemble high because each particle is associated with a …
    - Complete model state representation $\Rightarrow$ High memory need
    - ABM: Large populations & inter-agent interactions $\Rightarrow$ High computational burden
- Our lines of research: Active investigation with large-scale parallel (GPU, future: FPGA) & distributed computation

Agenda

- ✓Motivations & metaphors
- ✓Particle filtering "how" in a nutshell
- ✓Particle filtering: What
- ✓Particle filtering: How
- ✓Balancing model stochastics
- ✓Particle filtering with agent-based models
- Summary
- Supplemental material: Detailed documentation & derivations

Points of Notes

- Particle filtering continually regrounds model state given evidence from latest data
- With estimated current state, PF model can probabilistically project forward & be used for intervention evaluation
- Particle filtering is often far more effective than calibration, because of continual regrounding latent state
- Choice of likelihood function is very important
- Particle filtering can take many lines of evidence give a portrait of the underlying system and how it evolves
- Particle filter needs to balance
  - Too little confidence: Posterior distribution is too diffuse;model unable to predict even over short intervals
  - Too much confidence: Model does not lend enough credibility to observations, and gives poor & biased results
  - Model stochastics capture empirical stochastics & add humility
- Tuning probabilistic model parameters and stochastics makes a big difference for the accuracy of the particle filter

Conclusions

- Because of lack of strict distributional and model form assumptions, particle filtering is highly versatile
- Particle Filtering is well suited to work with many public health data streams & stochastic models
- In the presence of aggregate dynamic models, particle filtering can perform well
- Application of Particle Filtering is not a “turn the crank” process: it does involve iteration & learning
- Research progress is required to improve software support for Particle Filtering for ABM & DES models

Upcoming Events

Bootcamp & Incubator on Understanding Health Behavior using Smartphones & Wearables (June 24-26, 2019)

Detailed hands-on training in use of smartphone-based data collection for health. Instructors & TA assisted incubator to create, test, monitor, refine & analyze your study.

http://tinyurl.com/Smartphones4HealthBootcamp2019

Combining Data Science and Systems Science (Big Data and Dynamic Modeling) for Health (Jul 29-Aug 2, 2019)

Concrete guidance/resources for applying multiple concrete means of cross-leveraging data science and systems science

http://tinyurl.com/DataAndSystemScience2019

Agent-Based and Hybrid Modeling Bootcamp and Incubator for Health Researchers (Aug 19-24, 2019)

Hands-on learning for building agent-based and hybrid dynamic models. Incubator component provides close guidance and hands-on assistance building, testing and refining a working ABM.

http://tinyurl.com/ABMBootcamp2019

Agenda

- ✓Motivations & metaphors
- ✓Particle filtering “how” in a nutshell
- ✓Particle filtering: What
- ✓Particle filtering: How
- ✓Balancing model stochastics
- ✓Particle filtering with agent-based models
- ✓Summary
- Supplemental material: Detailed documentation & derivations

MORE DETAILED DERIVATION

State Space Modeling

- State space Model $\frac{dx^N}{dt} = g(x^N, \vartheta)$
  - N is the count of state variables
  - X^N represents a vector of length N
  - ϑ Represents noise in state evolution

State Space Modeling

- State space Model $\frac{dx^N}{dt} = g(x^N, \vartheta)$
  - N is the count of state variables
  - X^N represents a vector of length N (generally latent i.e., unobservable)
  - ϑ Represents noise in state evolution
- Unit updates: Solution for time t=k is:

$$x_k^N = g_k\left(x_{k-1}^N, \vartheta_{k-1}\right) = \int_{k-1}^{k} g(x^N(t), \vartheta)dt$$

- Where g_k advances the model from *k-1* to *k*

Simplifying Assumptions

- For ease of exposition, we make two assumptions here
  - Making these assumptions allows out exposition to be far simpler
  - These assumptions are for explanation and do not indicate limits to practical particle filter application
- Assumptions:
  - Measurements are made at regular intervals
  - The inter-measurement intervals are of unit length (i.e., measurements are separated by one time unit)
- Thus measurements occur at time k=1, k=2, etc., with measurement k occurring at time k.

Measurement Model

- We generally know a little bit about the underlying state of the system x_k^N at observation time *k* via noisy observations y_k^M
- The measurement model is as follows:

$$y_k^M = h_k(x_k^N, n_k)$$

where

- y_k^M represents a vector of length *M* giving the (noisy & partial) observations at time *k*
- n_k represents the noise affecting that observation
- h_k captures how state x_k^N is abstracted by observa.

Measurement Model

- We are seeking a way of estimating the state x_k^N at time k based on all of the observed data $y_{1:k}^M$
- The ongoing presence of noise throughout the process prevents naïve application of sampling via e.g., MCMC of the likelihood of observing $y_{1:k}^M$ for a sufficiently broad set of trajectories $x_{1:k}^N$
- To make this feasible, we seek a *recursive* way of estimating (sampling) x_k^N from x_{k-1}^N as a new observation y_k^M occurs
  - This mirrors classic Bayesian updating from a prior to a posterior when a new observation occurs
- This is updated in two stages: **Prediction** & **Update**

Recursive Update

- **Prediction**: The state x_{k-1}^N (sampled from p($x_{k-1}^N | y_{1:k-1}^M$)) at time k-1 to is mapped to samples from the state x_{k-}^N (sampled from p($x_k^N | y_{1:k-1}^M$)) at time *k just **prior to** the observation* y_k^M.
  - For our case (using the Condensation Algorithm), this update does not consider the coming observation y_k^M
- **Update**: x_{k-}^N is updated to x_k^N in a way that considers the current observation y_k^M

Dividing Our Difficulties

- Problem A: Theory for what needs to be sampled
  - This involves the theory for the distributions from which we are wishing to draw
- Problem B: Theory for how to do the sampling
  - This involves leveraging sequential importance sampling to actually perform the sampling in a viable way
- Problem C: Practicalities in performing the sampling
  - Reductions in sample diversity/effective sample size

Dividing Our Difficulties

- Problem A: Theory for what needs to be sampled
  - This involves the theory for the distributions from which we are wishing to draw
- Problem B: Theory for how to do the sampling
  - This involves leveraging sequential importance sampling to actually perform the sampling in a viable way
- Problem C: Practicalities in performing the sampling
  - Reductions in sample diversity/effective sample size

Prediction 1

- **Assumption:** p($x_{k-1}^{N} \mid y_{1:k-1}^{M}$) is available at time k-1
  - Note that p($x_0^N \mid y_0^M$) = p(x_0^N) is the *prior* for the entire particle filter.
  - We use this ★ to indicate this term in future slides
- Consider that for a binary outcome, B, if we can be confident of the identity:

$$p(A|C) = p(AB|C) + p(A\bar{B}|C)$$

- As formalized in the Chapman–Kolmogorov equation, this is further true for continuous system. Applying this principle, we can write

$$p\left(x_k^N \middle| y_{1:k-1}^M\right) = \int_{-\infty}^{+\infty} p\left(x_k^N, x_{k-1}^N \middle| y_{1:k-1}^M\right) dx_{k-1}^N$$

Prediction 2

- Consider the identity just shown:

$$p\left(x_k^N \middle| y_{1:k-1}^M\right) = \int_{-\infty}^{+\infty} p\left(x_k^N, x_{k-1}^N \middle| y_{1:k-1}^M\right) dx_{k-1}^N$$

- By the probabilistic chain rule, in general

$$p(AB|C) = \frac{p(ABC)}{P(C)} = \frac{p(A|BC)P(BC)}{P(C)} = p(A|BC)P(B|C)$$

- Thus we can rewrite the above as:

$$\int_{-\infty}^{+\infty} \mathrm{p}\left(x_k^N \middle| x_{k-1}^N, y_{1:k-1}^M\right) p\left(x_{k-1}^N \middle| y_{1:k-1}^M\right) dx_{k-1}^N$$

We call that in planning the recursive procedure,
We assumed (in the previous slide) that sampling from this term is possible at time k-1!

Prediction 3

- For a first-order Markov process such as this, the impact of past observations $y_{1:k-1}^{M}$ after time *k-1* is imparted purely through their influence on state x_{k-1}^{N}. Thus

$$p\left(x_k^N \middle| x_{k-1}^N, y_{1:k-1}^M\right) = p\left(x_k^N \middle| x_{k-1}^N\right)$$

Thus the previous formula

$$\int_{-\infty}^{+\infty} \mathrm{p}\left(x_k^N \middle| x_{k-1}^N, y_{1:k-1}^M\right) p\left(x_{k-1}^N \middle| y_{1:k-1}^M\right) dx_{k-1}^N$$

Can be rewritten as

$$p\left(x_k^N \middle| y_{1:k-1}^M\right) = \int_{-\infty}^{+\infty} p\left(x_k^N \middle| x_{k-1}^N\right) p\left(x_{k-1}^N \middle| y_{1:k-1}^M\right) dx_{k-1}^N$$

Given x_{k-1}^N, we sample from this by just simulating forward from time *k*-1 to time *k*!

These are just the samples available at time *k*-1 (once having processed the observation y_{k-1}^M)

Update 1

- The prediction phase provides the capacity to sample from $\mathrm{p}\left(x_k^N \middle| y_{1:k-1}^M\right)$
  - i.e., given y_{k-1}^M the prediction phase updates samples from $\mathrm{p}(x_{k-1}^N | y_{1:k-1}^M)$ to samples from $\mathrm{p}\left(x_k^N \middle| y_{1:k-1}^M\right)$
  - In future slides, we indicate this term $\mathrm{p}\left(x_k^N \middle| y_{1:k-1}^M\right)$ using
- The update phase maps samples from $\mathrm{p}\left(x_k^N \middle| y_{1:k-1}^M\right)$ to samples from $\mathrm{p}\left(x_k^N \middle| y_{1:k}^M\right)$
  - This is just updated from time *k*-1 to time *k*, which we indicate using the symbol
  - Here, we have just taken into account information from the observable y_k^M

Brief Summary of Where We're Headed

- Recall: The update phase maps samples from $\mathrm{p}\left(x_k^N \middle| y_{1:k-1}^M\right)$ to samples from $\mathrm{p}\left(x_k^N \middle| y_{1:k}^M\right)$
  - This is just  updated from time *k*-1 to time *k*, which we indicate using the symbol 

- Where we will get to is that sampling from simply requires sampling from a distribution where

$$p\left(x_k^N \middle| y_{1:k}^M\right) \propto p\left(y_k^M \middle| x_k^N\right) p\left(x_k^N \middle| y_{1:k-1}^M\right)$$

- **Warning**: A **very highly detailed** derivation lies ahead. If you are satisfied with the knowledge above, just "fast forward" ahead until after Slide "Update 10".

Update 2

- Recall: The update phase maps samples from $\mathrm{p}\left(x_k^N \middle| y_{1:k-1}^M\right)$ to samples from $\mathrm{p}\left(x_k^N \middle| y_{1:k}^M\right)$

- We can express

$$p\left(x_k^N \middle| y_{1:k}^M\right) = \frac{p(y_{1:k}^M | x_k^N) p(x_k^N)}{p(y_{1:k}^M)}$$

- Recognizing that $p\left(y_{1:k}^M\right) = p(y_{1:k-1}^M, y_k^M)$, we can rewrite this as:

$$p\left(x_k^N \middle| y_{1:k}^M\right) = \frac{p(y_{1:k-1}^M, y_k^M, | x_k^N) p(x_k^N)}{p(y_k^M, y_{1:k-1}^M)}$$

Update 3

- Consider previous result $p(x_k^N | y_{1:k}^M) = \frac{p(y_{1:k-1}^M, y_k^M, | x_k^N) p(x_k^N)}{p(y_k^M, y_{1:k-1}^M)}$
- By the probabilistic chain rule, in general

$$p(AB|C) = \frac{p(ABC)}{P(C)} = \frac{p(A|BC)P(BC)}{P(C)} = p(A|BC)P(B|C)$$

- Thus

$$p(y_{1:k-1}^M, y_k^M, | x_k^N) = p(y_{1:k-1}^M | y_k^M, x_k^N) p(y_k^M | x_k^N)$$

Update 4

- Consider previous result $p(x_k^N|y_{1:k}^M) = \frac{p(y_{1:k-1}^M, y_k^M, |x_k^N)p(x_k^N)}{p(y_k^M, y_{1:k-1}^M)}$
- By the probabilistic chain rule, in general

$$p(AB|C) = \frac{p(ABC)}{P(C)} = \frac{p(A|BC)P(BC)}{P(C)} = p(A|BC)P(B|C)$$

- Thus

$$p(y_{1:k-1}^M, y_k^M, |x_k^N) = p(y_{1:k-1}^M|y_k^M, x_k^N)p(y_k^M|x_k^N)$$

- Recall that by Bayes' rule: $p(A|B) = \frac{p(B|A)P(A)}{P(B)}$
- Thus $p(y_{1:k-1}^M|y_k^M, x_k^N) = \frac{p(y_k^M, x_k^N|y_{1:k-1}^M)p(y_{1:k-1}^M)}{p(y_k^M, x_k^N)}$
- Thus
- $p(y_{1:k-1}^M, y_k^M, |x_k^N) = \frac{p(y_k^M, x_k^N|y_{1:k-1}^M)p(y_{1:k-1}^M)}{p(y_k^M, x_k^N)} p(y_k^M|x_k^N)$

Update 5

- Recall from earlier that
  - $p(x_k^N | y_{1:k}^M) = \frac{p(y_k^M, y_{1:k-1}^M | x_k^N) p(x_k^N)}{p(y_k^M, y_{1:k-1}^M)}$
- Further recall that from the previous slide,
- $p(y_{1:k-1}^M, y_k^M, | x_k^N) = \frac{p(y_k^M, x_k^N | y_{1:k-1}^M) p(y_{1:k-1}^M)}{p(y_k^M, x_k^N)} p(y_k^M | x_k^N)$
- Putting the two together, we have
- $p(x_k^N | y_{1:k}^M) = \frac{p(y_k^M, x_k^N | y_{1:k-1}^M) p(y_{1:k-1}^M) p(y_k^M | x_k^N) p(x_k^N)}{p(y_k^M, x_k^N) p(y_k^M, y_{1:k-1}^M)}$

Update 6

- From the previous slide, we have

$$p(x_k^N|y_{1:k}^M) = \frac{p(y_k^M, x_k^N|y_{1:k-1}^M)p(y_{1:k-1}^M)p(y_k^M|x_k^N)p(x_k^N)}{p(y_k^M, x_k^N)p(y_k^M, y_{1:k-1}^M)}$$

- Recall further that by the definition of a joint distribution,

$$p(y_k^M, y_{1:k-1}^M) = p(y_k^M | y_{1:k-1}^M)p(y_{1:k-1}^M)$$

- Substituting in for $p(y_k^M, y_{1:k-1}^M)$, we get

$$p(x_k^N|y_{1:k}^M) = \frac{p(y_k^M, x_k^N|y_{1:k-1}^M)p(y_{1:k-1}^M)p(y_k^M|x_k^N)p(x_k^N)}{p(y_k^M, x_k^N)p(y_k^M | y_{1:k-1}^M)p(y_{1:k-1}^M)}$$

- Cancelling the term involving $p(y_{1:k-1}^M)$, we have

$$p(x_k^N|y_{1:k}^M) = \frac{p(y_k^M, x_k^N|y_{1:k-1}^M)p(y_k^M|x_k^N)p(x_k^N)}{p(y_k^M, x_k^N)p(y_k^M | y_{1:k-1}^M)}$$

Update 7

- From the previous slide, we have

$$p\left(x_k^N \middle| y_{1:k}^M\right) = \frac{p\left(y_k^M, x_k^N \middle| y_{1:k-1}^M\right) p\left(y_k^M \middle| x_k^N\right) p\left(x_k^N\right)}{p\left(y_k^M, x_k^N\right) p\left(y_k^M \middle| y_{1:k-1}^M\right)}$$

- We can now further recognize that

$$p\left(y_k^M, x_k^N\right) = p\left(y_k^M \middle| x_k^N\right) p\left(x_k^N\right)$$

- And by cancelling reduce the above to

$$p\left(x_k^N \middle| y_{1:k}^M\right) = \frac{p\left(y_k^M, x_k^N \middle| y_{1:k-1}^M\right) p\left(y_k^M \middle| x_k^N\right) p\left(x_k^N\right)}{p\left(y_k^M \middle| x_k^N\right) p\left(x_k^N\right) p\left(y_k^M \middle| y_{1:k-1}^M\right)} = \frac{p\left(y_k^M, x_k^N \middle| y_{1:k-1}^M\right)}{p\left(y_k^M \middle| y_{1:k-1}^M\right)}$$

Update 8

- From the previous slide, we have

$$p\left(x_k^N \middle| y_{1:k}^M\right) = \frac{p\left(y_k^M, x_k^N \middle| y_{1:k-1}^M\right)}{p\left(y_k^M \middle| y_{1:k-1}^M\right)}$$

- Recall that by the probabilistic chain rule, in general

$$p(AB|C) = \frac{p(ABC)}{P(C)} = \frac{p(A|BC)P(BC)}{P(C)} = p(A|BC)P(B|C)$$

- Thus we have

$$p\left(x_k^N \middle| y_{1:k}^M\right) = \frac{p\left(y_k^M, x_k^N \middle| y_{1:k-1}^M\right)}{p\left(y_k^M \middle| y_{1:k-1}^M\right)}$$

$$= \frac{p\left(y_k^M \middle| x_k^N, y_{1:k-1}^M\right) p\left(x_k^N \middle| y_{1:k-1}^M\right)}{p\left(y_k^M \middle| y_{1:k-1}^M\right)}$$

Update 9

- From the previous slide, we have

$$p\left(x_k^N \middle| y_{1:k}^M\right) = \frac{p\left(y_k^M \middle| x_k^N, y_{1:k-1}^M\right) p\left(x_k^N \middle| y_{1:k-1}^M\right)}{p\left(y_k^M \middle| y_{1:k-1}^M\right)}$$

- Recall again that for a first-order Markov process such as this, the impact of past observations $y_{1:k-1}^M$ after time *k-1* is imparted purely through their influence on state x_{k-1}^N.
- By the properties of a first-order Markov chain, we know that the state

$$p\left(y_k^M \middle| x_k^N, y_{1:k-1}^M\right) = p\left(y_k^M \middle| x_k^N\right)$$

- Where $p\left(y_k^M \middle| x_k^N\right)$ is the likelihood of observing y_k^M given x_k^N, and which we indicate in our formulae using
- Thus, in summary, we have

$$p\left(x_k^N \middle| y_{1:k}^M\right) = \frac{p\left(y_k^M \middle| x_k^N\right) p\left(x_k^N \middle| y_{1:k-1}^M\right)}{p\left(y_k^M \middle| y_{1:k-1}^M\right)}$$

Update 10

- From the previous slide we have

$$p(x_k^N | y_{1:k}^M) = \frac{p(y_k^M | x_k^N) p(x_k^N | y_{1:k-1}^M)}{p(y_k^M | y_{1:k-1}^M)}$$

- Particularly because our expression will in general not be expressible in closed form, our interest here lies in **sampling from** $p(x_k^N | y_{1:k}^M)$.
- Because the denominator

$$p(y_k^M | y_{1:k-1}^M) = \int_{-\infty}^{+\infty} p(y_k^M | X_k^N) p(X_k^N | y_{1:k-1}^M) dX_k^N$$

- does not depend on x_k^N, performing the sampling does not require calculating it. Thus it is sufficient to sample using the knowledge that

$$p(x_k^N | y_{1:k}^M) \propto p(y_k^M | x_k^N) p(x_k^N | y_{1:k-1}^M)$$

Sampling from the full trajectory

- Recall from "Update 10" that

$$p\left(x_k^N \middle| y_{1:k}^M\right) \propto p(y_k^M | x_k^N) p(x_k^N | y_{1:k-1}^M)$$

By extension, we have

$$p\left(x_{0:k}^N \middle| y_{1:k}^M\right) \propto p(y_k^M | x_{0:k}^N) p(x_{0:k}^N | y_{1:k-1}^M)$$

Recognizing that $x_{0:k}^N$ is composed of pieces, we rewrite this as

$$p(y_k^M | x_{0:k}^N) p(x_k^N, x_{0:k-1}^N | y_{1:k-1}^M)$$

By the probabilistic chain rule, in general

$$p(AB|C) = \frac{p(ABC)}{P(C)} = \frac{p(A|BC)P(BC)}{P(C)} = p(A|BC)P(B|C)$$

Thus we can rewrite the above as

$$p\left(x_{0:k}^N \middle| y_{1:k}^M\right)$$
$$\propto p\left(y_k^M \middle| x_k^N, x_{0:k-1}^N\right) p\left(x_k^N \middle| x_{0:k-1}^N, y_{1:k-1}^M\right) p(x_{0:k-1}^N | y_{1:k-1}^M)$$

Full Trajectory Distribution

- From the previous slide, we have
$$p\left(x_{0:k}^{N}\middle|y_{1:k}^{M}\right) \propto p\left(y_{k}^{M}\middle|x_{k}^{N},x_{0:k-1}^{N}\right)p\left(x_{k}^{N}\middle|x_{0:k-1}^{N},y_{1:k-1}^{M}\right)p\left(x_{0:k-1}^{N}\middle|y_{1:k-1}^{M}\right)$$
- because we assume a first-order Markov process, we recognize that two terms above can be simplified
$$p\left(y_{k}^{M}\middle|x_{k}^{N},x_{0:k-1}^{N}\right) = p\left(y_{k}^{M}\middle|x_{k}^{N}\right)$$
$$p\left(x_{k}^{N}\middle|x_{0:k-1}^{N},y_{1:k-1}^{M}\right) = p\left(x_{k}^{N}\middle|x_{k-1}^{N}\right)$$
- Substituting these in above, we have
$$p\left(x_{0:k}^{N}\middle|y_{1:k}^{M}\right) \propto p\left(y_{k}^{M}\middle|x_{k}^{N}\right)p\left(x_{k}^{N}\middle|x_{k-1}^{N}\right)p\left(x_{0:k-1}^{N}\middle|y_{1:k-1}^{M}\right)$$

Dividing Our Difficulties

- Problem A: Theory for what needs to be sampled
  - This involves the theory for the distributions from which we are wishing to draw
- Problem B: Theory for how to do the sampling
  - This involves leveraging sequential importance sampling to actually perform the sampling in a viable way
- Problem C: Practicalities in performing the sampling
  - Reductions in sample diversity/effective sample size

Taking Stock

- The previous section indicated the “target distribution” from which have to sample
- This is a distribution $p\left(x_{0:k}^{N} \middle| y_{1:k}^{M}\right)$ over trajectories $x_{0:k}^{N}$ of latent states of the system, given observations $y_{1:k}^{M}$
- This section examines the mathematics underlying the scheme for *how we actually go about sampling from that distribution*

Common Problem: Sampling Difficulty

- Two common ways of sampling from a distribution are
  - Computing the cumulative distribution and then using a draw from 1 or more uniform distributions to sample from the cumulative
  - Sampling via MCMC
- There are sometimes problems with such approaches
  - Lack of closed-form specification of the distribution $\Rightarrow$ Cannot derive closed-form characterization of the cumulative distribution
  - The distribution has extremely high dimensionality $\Rightarrow$ Difficult to use MCMC and related techniques, compute cumulative distribution function
    - **This applies to our case**

Reflection: Why MCMC Based Sampling Won't Work with Stochastic Models

- So we have to sample from a (recursively defined) distribut.
- A well-established traditional way for sampling parameters – including with dynamic models – is to use MCMC
- Two central difficulty for doing this for the latent state with a stochastic dynamic model is that
  - We have to simple from a massive number of different values – values stochastically evolving at each time step
  - In contrast to the case with MCMC (where we are sampling from parameter values that drive the dynamic model), the latent state of the model is emergent *from* the dynamic model
    - We thus can't simply sample different values of it and assess the likelihood of each based on the output
    - In order to try to sample these latent states via MCMC, we would need to generate again and again, without a clear ability to sample from high density regions $\Rightarrow$ Great inefficiency

Another Means of Sampling

- Particle Filtering uses another means of sampling – sequential importance sampling
- Here, we forsake the infeasible goal of sampling directly from the ideal ("target") distribution via MCMC
- Rather, we use the importance sampling technique of
  - Sampling readily from a different "proposal distribution"
  - Using weighting of the samples from the proposal distribution to characterize their relative representation in the target relative to the proposal distribution
  - This weight allows us to sample more frequently from those samples that are well represented in the target distribution, and less frequently from others
- The coming slides present the theory behind importance sampling & sequential importance sampling

Importance Sampling 1

- Suppose that we want to draw samples from a “target” distribution p(x) that
  - Is difficult to sample from directly
  - Given an x, has a value that can be readily computed
- Suppose that we have another “proposal” distribution q(x) from which we can readily sample
  - Common example: Uniform distribution
  - Ideally this would be something like p(x)

Importance Sampling 2

- We can readily sample *N* values from p(x) by a 4 part procedure:
  - Part 1: Create a set S of N sample values x_i ($1 \le i \le N$) from q(x)
    - e.g., draw each x_i from a uniform distribution between 0 and 1
  - Part 2: Label each drawn value x_i with a "weight" $W_i = p(x_i)/q(x_i)$
    - This "weight" expresses how much more common x_i is within the target distribution p(x), when compared to the distribution from which it was drawn, q(x)
  - Part 3: Normalize weights, labeling each x_i with weight $w_i = \frac{W_i}{\sum_{j=1}^{N} W_j}$
  - Part 4: Draw N samples from S, where the probability of drawing sample i each is proportional to w_i
    - For each such sample, this is readily performed by
      - Drawing a value *u* from [0,1]
      - Going through each $1 \le i \le N$ accumulating the value of w_i until the smallest i where $\sum_{j=1}^{i} W_j \ge u$

Importance Sampling 3

- Importance sampling approximation to p(x) is as follows:

$$\hat{p}(x) \approx \sum_{i=1}^{n} w(x^{(i)}) \delta_{X^{(i)}}$$

Where the normalized weights $w\left(x^{(i)}\right)$ are given by

$$w\left(x^{(i)}\right) = \frac{W\left(x^{(i)}\right)}{\sum_{j=1}^{N} W(x^{(j)})}$$

Based on unnormalized weights $W\left(x^{(i)}\right)$:

$$W\left(x^{(i)}\right) = \frac{p(x^{(i)})}{q(x^{(i)})}$$

And $\delta_{X^{(i)}}$ is the Dirac Delta probability mass located at point $x^{(i)}$

Sequential Importance Sampling for Our Case

- Importance-weighted samples are maintained over time
  - Such samples are termed “particles”
- Successive observations at integer times are made, with each such observation updating the underlying distribution $\mathrm{p}\left(x_{0:k}^{N(i)} \middle| y_{1:k}^{M}\right)$
  - Direct sampling from this distribution is not generally possible
- To enable successively sample from $\mathrm{p}\left(x_{0:k}^{N(i)} \middle| y_{1:k}^{M}\right)$ as each observations *k* arrives, we draw instead from a proposal distribution *q(x)* & successively update samples $x_{1:k-1}^{N(i)}$ & weights $w_{k-1}^{(i)}$ to reflect the observation, yielding $x_{1:k}^{N(i)}$ and $w_{k}^{(i)}$

Importance Sampling to Draw From

- Target distribution is $p\left(x_{0:k-1}^{N(i)} \middle| y_{1:k-1}^{M}\right)$
- $x_{1:k-1}^{N(i)}$ is drawn from proposal distribution $q\left(x_{0:k-1}^{N(i)} \middle| y_{1:k-1}^{M}\right)$
- Here we are seeking to choose and maintain a *proposal distribution* that can be readily sampled from over time
  - Per importance sampling principles, these samples are then weighted as

$$w_{k-1}^{(i)} = \frac{p\left(x_{0:k-1}^{N(i)} \middle| y_{1:k-1}^{M}\right)}{q\left(x_{0:k-1}^{N(i)} \middle| y_{1:k-1}^{M}\right)}$$

Convenient Choice in Proposal Distribution

- *For convenience, we assume that*

$$q\left(x_{1:k}^{N} \middle| y_{1:k}^{M}\right) = q\left(x_{k}^{N} \middle| x_{1:k-1}^{N}, y_{1:k}^{M}\right) q\left(x_{1:k-1}^{N} \middle| y_{1:k-1}^{M}\right)$$

- These terms represent
  - $q\left(x_{1:k-1}^{N} \middle| y_{1:k-1}^{M}\right)$ is just the value for the proposal distribution at the time of the previous observations
  - $x_{k}^{N(i)} \sim q(x_{k}^{N} | x_{1:k-1}^{N}, y_{1:k}^{M})$: the probability (density) of the updated state taking into account the new measurement measurements at time *k*

Key Need: Weight Updates

- Given: A sample (particle) drawn with a weight $w_{k-1}^{(i)}$ from the importance-sampling approximated distribution $p\left(x_{1:k-1}^{N} \middle| y_{1:k-1}^{M}\right)$ at time *k-1*
- Need: Formulate an equation to *update the weight of that particle* to draw from the target distribution $p\left(x_{0:k}^{N(i)} \middle| y_{1:k}^{M}\right)$ for time. This update would take into account both
  - The model-borne dynamics mapping from x_{k-1}^{N} to x_{k}^{N}
  - The observation y_{k}^{M}

- Recall from earlier that we have

$$w_k^{(i)} = \frac{p\left(x_{0:k}^{N(i)} \middle| y_{1:k}^{M}\right)}{q\left(x_{0:k}^{N(i)} \middle| y_{1:k}^{M}\right)}$$

- Recall from earlier slide "Full Trajectory Distribution" that

$$p\left(x_{0:k}^{N} \middle| y_{1:k}^{M}\right) \propto p\left(y_k^{M} \middle| x_k^{N}\right) p\left(x_k^{N} \middle| x_{k-1}^{N}\right) p\left(x_{0:k-1}^{N} \middle| y_{1:k-1}^{M}\right)$$

- Recall further that we chose a proposal distribution of the form

$$q\left(x_{1:k}^{N} \middle| y_{1:k}^{M}\right) = q\left(x_k^{N} \middle| x_{1:k-1}^{N}, y_{1:k}^{M}\right) q\left(x_{1:k-1}^{N} \middle| y_{1:k-1}^{M}\right)$$

- We thus have

$$w_k^{(i)} = \frac{p\left(y_k^{M} \middle| x_k^{N}\right) p\left(x_k^{N} \middle| x_{k-1}^{N}\right) p\left(x_{0:k-1}^{N} \middle| y_{1:k-1}^{M}\right)}{q\left(x_k^{N} \middle| x_{1:k-1}^{N}, y_{1:k}^{M}\right) q\left(x_{1:k-1}^{N} \middle| y_{1:k-1}^{M}\right)}$$

Recursive Weight Updates

- Recall from the previous slide that

$$w_k^{(i)} = \frac{p\left(y_k^M \middle| x_k^N\right) p\left(x_k^N \middle| x_{k-1}^N\right)}{q\left(x_k^N \middle| x_{1:k-1}^N, y_{1:k}^M\right)} \frac{p\left(x_{0:k-1}^N \middle| y_{1:k-1}^M\right)}{q\left(x_{1:k-1}^N \middle| y_{1:k-1}^M\right)}$$

- Recalling from a few slides earlier that

$$w_{k-1}^{(i)} = \frac{p\left(x_{0:k-1}^{N(i)} \middle| y_{1:k-1}^M\right)}{q\left(x_{0:k-1}^{N(i)} \middle| y_{1:k-1}^M\right)}$$

- We can recognize the ratio of the final terms in the numerator & denominator as $w_{k-1}^{(i)}$, and thus rewrite the equation as:

$$w_k^{(i)} = \frac{p\left(y_k^M \middle| x_k^N\right) p\left(x_k^N \middle| x_{k-1}^N\right)}{q\left(x_k^N \middle| x_{1:k-1}^N, y_{1:k}^M\right)} w_{k-1}^{(i)}$$

Recursive Weight Updates Redux

- From the previous slide, we have

$$w_k^{(i)} = \frac{p\left(y_k^M \middle| x_k^N\right) p\left(x_k^N \middle| x_{k-1}^N\right)}{q\left(x_k^N \middle| x_{1:k-1}^N, y_{1:k}^M\right)} w_{k-1}^{(i)}$$

- Because the process g_k is first-order Markovian, we can further simplify the denominator by recognizing that all impact of past state is captured in the previous state $x_{k-1}^{N(i)}$, and by assuming that all observations prior to k have already between reflected in the distribution, thus:

$$q\left(x_k^{N(i)} \middle| x_{1:k-1}^{N(i)}, y_{1:k}^M\right) = q\left(x_k^{N(i)} \middle| x_{k-1}^{N(i)}, y_k^M\right)$$

and (rearranging slightly) we have

$$w_k^{(i)} \propto w_{k-1}^{(i)} \frac{p\left(y_k^M \middle| x_k^{N(i)}\right) p\left(x_k^{N(i)} | x_{k-1}^{N(i)}\right)}{q\left(x_k^{N(i)} \middle| x_{k-1}^{N(i)}, y_k^M\right)}$$

Naïve Algorithm

Initialization (k=0)

- Sample $X_0^{N(i)}$ from $q_0(x_0^N | y_0^M)$
- *For each particle i*
  - $w_k^{(i)} = \frac{p(X_0^{N(i)}) p(y_0^M | X_0^{N(i)})}{q(X_0^{N(i)} | y_0^M)}$
- For each particle i
  - $W_0^{(i)} = \frac{w_0^{(i)}}{\sum_{i=1}^{N_S} w_0^{(i)}}$

Ongoing Observations (k>0)

- For each particle i
  - Advance state via sampling $X_k^{N(i)} \sim q(x_k^N | y_k, X_{0:k-1}^{N(i)})$
  - Supplement trajectory $X_{0:k}^{N(i)} = (X_{0:k-1}^{N(i)}, X_k^{N(i)})$
  - Update weight

$$w_k^{(i)} = w_{k-1}^{(i)} \frac{p\left(y_k^M \middle| x_k^{N(i)}\right) p\left(x_k^{N(i)} | x_{k-1}^{N(i)}\right)}{q\left(x_k^{N(i)} \middle| x_{k-1}^{N(i)}, y_k^M\right)}$$

- For each particle i

$$W_k^{(i)} = \frac{w_k^{(i)}}{\sum_{i=1}^{N_s} w_k^{(i)}}$$

Practical Problem: Reduced Effective Sample Size

- Performed naively, the algorithm above can lead to a situation where
  - Most particles have very low weight
  - Only a few particles have significant weight
- This situation gives a low "effective sample size" in that with sequential importance sampling, the high weight particles will be overwhelmingly overrepresented
- We can recognize this situation by monitoring the variance in weights, using the second moment of the weights

$$S_{eff} = \frac{1}{\sum_{i=1}^{N} (w_k^{(i)})^2}$$

- If $S_{eff} < S_T$, we view the samples as suffering too low a sample size

Resampling Step

- When there is too large a diversity of particle weights following the updating of weights, we *perform a (weighted) resampling* from the particles
  - We draw a new set of particles from the set of particles (reflecting the updated weights), where the chance of selecting a given particle is proportional to is weight
  - A given particle may be disappear or be duplicated many times
    - NB: If it is duplicated several times, note that the resulting particles have a complete copy of the state of the original particle (such that this state can then evolve independently)
  - The resampled particles are assigned a weight of 1

Resampling Step

- *Suppose we have particles* $X_k^{N(i)}$ (call these $X_{k-}^{N(i)}$) whose weight $W_{k-}^{(i)}$ has just been updated by an observation;
- Let $S_{eff} = \frac{1}{\sum_{i=1}^{N}(W_k^{(i)})^2}$
- If $S_{eff} < S_T$, then $\forall i, 1 \leq i \leq N$
  - $X_{k+}^{N(i)} \sim X_{k-}^{N(multinomial(W_{k-}^{(1..N)}))}$
  - $W_{k+}^{(i)} = 1/N$
- Please note that $multinomial(W_{k-}^{(1..N)})$ represents a sample from the multinomial distribution, taking as arguments N parameters representing the probabilities of returning each value (here $W_{k-}^{(1..N)}$) and returning an index of the chosen value

Algorithm Maintaining Sample Size

Initialization (k=0)

- Sample $X_0^{N(i)}$ from $q_0(x_0^N | y_0^M)$
- *For each particle i*
  - $w_k^{(i)} = \frac{p(X_0^{N(i)}) p(y_0^M | X_0^{N(i)})}{q(X_0^{N(i)} | y_0^M)}$
- For each particle i
  - $W_0^{(i)} = \frac{w_0^{(i)}}{\sum_{i=1}^{N_S} w_0^{(i)}}$

Ongoing Observations (k>0)

- For each particle i
  - Advance state via sampling $X_k^{N(i)} \sim q(x_k^N | y_k, X_{0:k-1}^{N(i)})$
  - Supplement trajectory $X_{0:k}^{N(i)} = (X_{0:k-1}^{N(i)}, X_k^{N(i)})$
  - Update weight

$$w_k^{(i)} = w_{k-1}^{(i)} \frac{p\left(y_k^M \middle| x_k^{N(i)}\right) p\left(x_k^{N(i)} \middle| x_{k-1}^{N(i)}\right)}{q\left(x_k^{N(i)} \middle| x_{k-1}^{N(i)}, y_k^M\right)}$$

- For each particle i

$$W_k^{(i)} = \frac{w_k^{(i)}}{\sum_{i=1}^{N_S} w_k^{(i)}}$$

- Let $S_{eff} = \frac{1}{\sum_{i=1}^{N_S} (W_k^{(i)})^2}$
- If $S_{eff} < S_T$, resample $X_k^{N(i)}$

Choice of Proposal Distribution

- The algorithms above leave open to the implementer the choice of a proposal distribution

$$q(x_k^N | y_k, X_{0:k-1}^{N(i)})$$

- The choice of proposal distribution can have a sizeable impact on
  - The practical performance of the particle filtering
  - The complexity of the implementation
- Many practitioners make use of the “condensation algorithm” of Isard & Blake (1998), which employs a particularly simple proposal distribution

The "*condensation algorithm*" *(Isard & Blake 1998)*

- Here, one choses $q\left(x_k^{N(i)}\middle|x_{k-1}^{N(i)},y_k^M\right)=p(x_k^{N(i)}|x_{k-1}^{N(i)})$
  - In other words, the **proposal** distribution simply uses the simulation induced probability distribution and ignores the data
- Recall that, in general, for particle *i*
  - $w_k^{(i)}=w_{k-1}^{(i)}\frac{p\left(y_k^M\middle|x_k^{N(i)}\right)p(x_k^{N(i)}|x_{k-1}^{N(i)})}{q\left(x_k^{N(i)}\middle|x_{k-1}^{N(i)},y_k^M\right)}$
- For $q\left(x_k^{N(i)}\middle|x_{k-1}^{N(i)},y_k^M\right)=p(x_k^{N(i)}|x_{k-1}^{N(i)})$:

$$w_k^{(i)}=w_{k-1}^{(i)}\frac{p\left(y_k^M\middle|x_k^{N(i)}\right)p(x_k^{N(i)}|x_{k-1}^{N(i)})}{p(x_k^{N(i)}|x_{k-1}^{N(i)})}=w_{k-1}^{(i)}p\left(y_k^M\middle|x_k^{N(i)}\right)$$

- Recall further that$p\left(y_k^M\middle|x_k^{N(i)}\right)$ is the **likelihood** of observing the empirical data at that time (y_t) given the particle state $x_k^{N(i)}$ at that time