Estimation of parameters for stochastic dynamic models

Ben Bolker

McMaster University
Departments of Mathematics & Statistics and Biology

20 June 2013

Outline

Outline

Modeling

Typical stats	Typical math
stochastic	deterministic
static	dynamic
phenomenological	mechanistic

Standard time-series models (ARIMA, spectral/wavelet analyses) are (mostly) phenomenological

Process and measurement error

- For stochastic models need to define both a **process model** and an **observation model** (= measurement model)

  Process model $Y(t+1) \sim F(Y(t))$

  Measurement model $Y_{\text{obs}}(t) \sim Y(t)$

- Only process error affects the future dynamics of the process (usually)
- Might decompose process model into a deterministic model for the expectation and (additive?) noise around the expectation: e.g. $Y(t) = \mu + \epsilon$, $Y(t) \sim \text{Poisson}(\exp(\eta))$

Process and measurement error

- For stochastic models need to define both a **process model** and an **observation model** (= measurement model)

  Process model $Y(t+1) \sim F(Y(t))$

  Measurement model $Y_{\text{obs}}(t) \sim Y(t)$

- Only **process** error affects the future dynamics of the process (usually)
- Might decompose process model into a deterministic model for the expectation and (additive?) noise around the expectation: e.g. $Y(t) = \mu + \epsilon$, $Y(t) \sim \text{Poisson}(\exp(\eta))$

Process and measurement error

- For stochastic models need to define both a **process model** and an **observation model** (= measurement model)

  Process model $Y(t+1) \sim F(Y(t))$

  Measurement model $Y_{\text{obs}}(t) \sim Y(t)$
- Only **process** error affects the future dynamics of the process (usually)
- Might decompose process model into a deterministic model for the expectation and (additive?) noise around the expectation: e.g. $Y(t) = \mu + \epsilon$, $Y(t) \sim \text{Poisson}(\exp(\eta))$

Consequences

- Process error induces dynamic **changes in variance**
- Process+observation error induce **correlations** between subsequent observations
- Observation at next time step depends on **unobserved** value at current time step
- Simple statistical methods
(i.e. uncorrelated, equal variance)
are incorrect

Consequences

- Process error induces dynamic **changes in variance**
- Process+observation error induce **correlations** between subsequent observations
- Observation at next time step depends on **unobserved** value at current time step
- Simple statistical methods (i.e. uncorrelated, equal variance) are incorrect

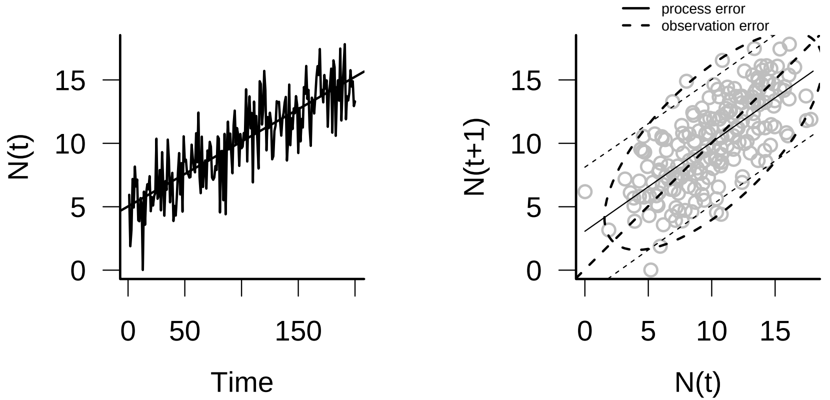
Consequences

- Process error induces dynamic **changes in variance**
- Process+observation error induce **correlations** between subsequent observations
- Observation at next time step depends on **unobserved** value at current time step
- Simple statistical methods (i.e. uncorrelated, equal variance) are incorrect

Consequences

- Process error induces dynamic **changes in variance**
- Process+observation error induce **correlations** between subsequent observations
- Observation at next time step depends on **unobserved** value at current time step
- Simple statistical methods
  (i.e. uncorrelated, equal variance)
  are incorrect

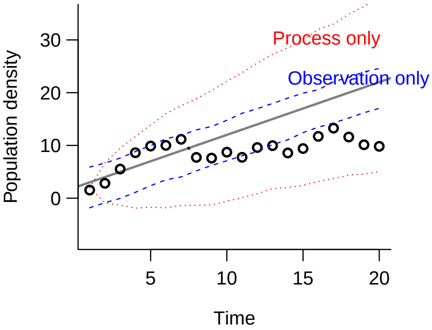
Linear example



How should we interpret this single realization?

Outline

Linear model



$$N(1) = a$$
$$N(t+1) \sim \text{Normal}(N(t) + b, \sigma^2_{\text{proc}})$$
$$N_{\text{obs}}(t) \sim \text{Normal}(N(t), \sigma^2_{\text{obs}})$$

R code (version 1)

```
## set up parameters etc.
nt <- 20; a <- 6; b <- 1
sd_proc <- sqrt(2)
sd_obs <- sqrt(2)
N <- Nobs <- numeric(nt)
set.seed(101)  ## for reproducibility
## actual model
N[1] <- a
Nobs[1] <- rnorm(1,N[1],sd_obs)
for (i in 1:nt) {
  N[i+1] <- rnorm(1,N[i]+b,sd_proc)
  Nobs[i+1] <- rnorm(1,N[i+1],sd_proc)
}
```

R code (version 2)

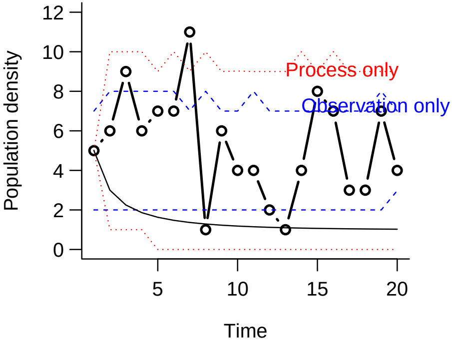
```
library("deSolve")

##
## Attaching package:  'deSolve'
##
## The following object is masked from
'package:graphics':
##
##     matplot

linfun <- function(t,y,parms) {
  g <- with(as.list(c(y,parms)), {
     N_new <- rnorm(1,N+b,sd_proc)
     c(N=N_new,Nobs=rnorm(1,N_new,sd_obs))
```

R code (version 3)

For this particular example, we can cheat because the process error doesn't really affect the future dynamics — it just accumulates:

```
N_det <- a+b*(0:(nt-1))
set.seed(101)  ## for reproducibility
N <- N_det+cumsum(c(0,rnorm(nt-1,0,sd_proc)))
N_obs <- rnorm(nt,N,sd_obs)
```

Hyperbolic nonlinear model



$$\begin{aligned} N(1) &= N_0 \\ N(t+1) &\sim \text{Poisson}(aN(t)/(b+N(t))) \\ N_{\text{obs}}(t) &\sim \text{Binomial}(N(t), p) \end{aligned}$$

(Equating (1) process-error-only and observation-error-only and (2) deterministic and stochastic version is fairly hard ...)

Stochastic ODEs

- continuous-time, continuous-state
- ordinary differential equations plus a **Wiener process** (= derivative of a Brownian motion)
- delicate analysis (For biologists: Turelli (1977); Roughgarden (1995). For mathematicians: Øksendal (2003))
- Specialized integration methods
- Better for cellular/physiological than population models?

Markov processes

- continuous-time, discrete-state
- specify (limits of) probabilities of transitions per unit time, e.g. $P(N \rightarrow N+1)$ in the interval $(t, t+dt)$ is $rN(t)\,dt$
- Even harder than SDEs to analyze rigorously ...
- But computationally straightforward: **Gillespie algorithm** and variations (Gillespie, 2007): exponentially distributed time between transitions

Outline

Trajectory matching

- Easiest approach: just simulate the deterministic version of the model (i.e., with neither observation nor process error) and compare
- Because measurement error is (typically) independent at each observation, just have to multiply probabilities/add log-likelihoods
- for Normally distributed, equal-variance error, maximum likelihood estimation is equivalent to OLS fitting
- Very common for ODE model fitting, e.g. Gani and Leach (2001); van Veen et al. (2005).

Pseudo-code

```
## deterministic dynamics:
## function of parameters, possibly including ICs
determ_fun <- function(determ_params) { ... }
## objective function (neg. log-likelihood, SSQ, ...)
## 'params' includes process and observation parameters
obj_fun <- function(params,data) {
  estimate <- determ_fun(params[determ_params])
  obj <- fun(estimate,data,params[obs_params])
  return(obj)
}
find_minimum(obj_fun,starting_params,...)
```

Real code (using for loops)

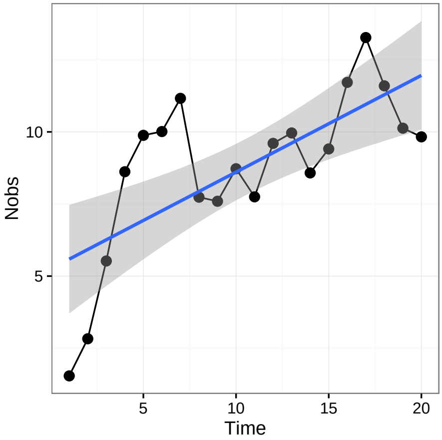
```
determ_fun <- function(p,nt) {
  with(as.list(p),a+b*(1:nt))
}
obj_fun <- function(p,nt,Nobs) {
  estimate <- determ_fun(p[c("a","b")],nt)
  ## negative log-lik. of Normal
  obj <- -sum(dnorm(Nobs,estimate,p["sd"],log=TRUE))
  return(obj)
}
optim(fn=obj_fun,par=c(a=5,b=2,sd=1),nt=20,Nobs=linN)
```

Real code (using mle2())

```
library(bbmle)
determ_fun <- function(a,b,nt) a+b*(1:nt)
mle2(Nobs~dnorm(determ_fun(a,b,nt),sd),
     data=list(Nobs=linN,nt=nt),
     start=list(a=5,b=2,sd=1),
     method="Nelder-Mead")

## Warning in calc_mle2_function(minuslogl,
parameters, start = start, parnames = parnames, :
using dnorm() with sd implicitly set to 1 is rarely
sensible
```

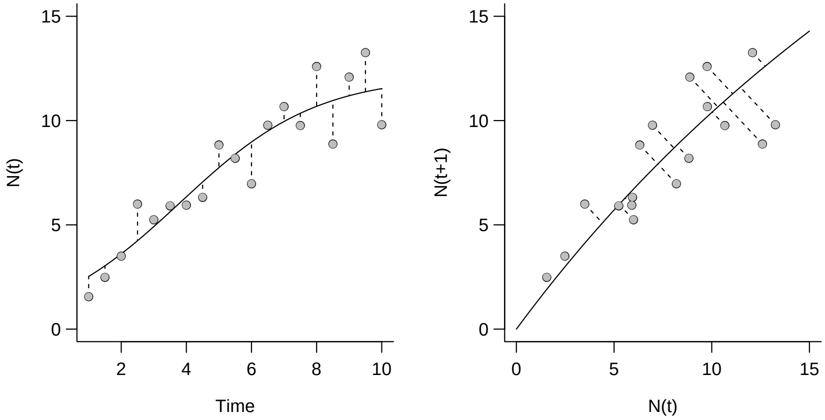
mle2() simplifies computation of confidence intervals, likelihood profiles, predicted values, etc.

Real code (using linear regression, `lm()`)




```
linDF <- data.frame(Time=1:nt,
                    Nobs=linN)
lm(Nobs~Time,data=linDF)
```

Logistic model fit



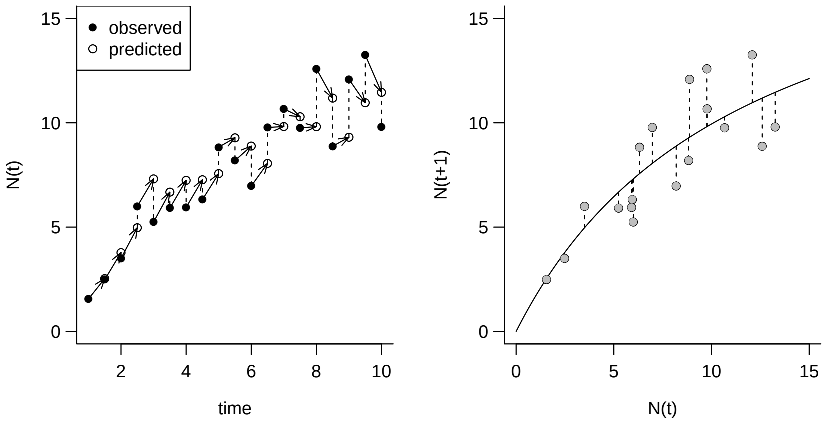
Gradient matching

- Next-easiest approach: assume **only** process error (no measurement error)
- $N(t+1)$ depends only on $N(t)$ (which we know exactly): **conditional independence**
- **One-step-ahead prediction**
- Simple for discrete-time models
  (we need to specify $N(t+1) \sim N(t)$ anyway)
- Somewhat more complicated for continuous-time models (Ellner et al., 2002)

Pseudo-code

```
## deterministic dynamics:
## function of parameters and previous values
onestep_fun <- function(determ_params,Nt) { ... }
## objective function (neg. log-likelihood, SSQ, ...)
obj_fun <- function(params,data) {
  obj <- ... ## numeric vector of length (nt-1)
  for (i in 1:(nt-1)) {
     estimate <- onestep_fun(N[i],params[determ_params]))
     obj[i] <- fun(estimate,N[i+1],params[obs_params])
  }
  return(sum(obj))
}
find_minimum(obj_fun,starting_params,...)
```

Logistic growth fit

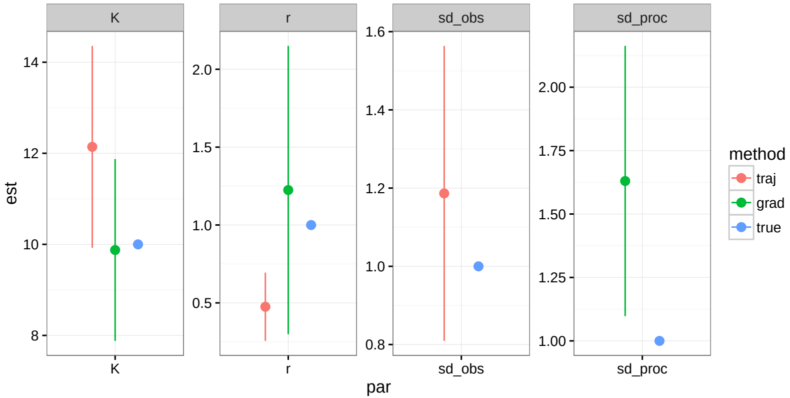


Comparison

How can we use these?

- Try both and hope the answers are not importantly different …
- Use biological knowledge of whether process $\gg$ observation error or vice versa

Logistic fit comparisons



Outline

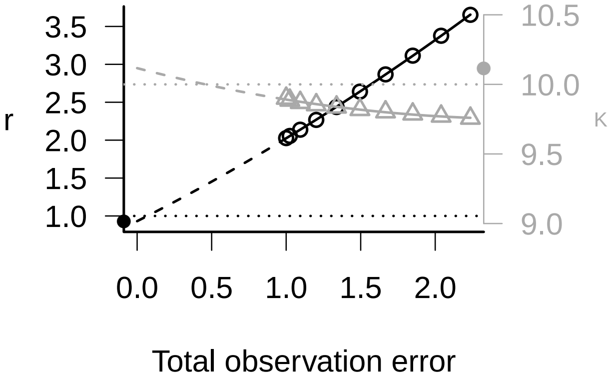
SIMEX

- **SIM**ulation-**EX**trapolation method
- Requires (1) an independent estimate of the observation error; (2) that we can sensibly add **additional** observation error to the data
- Slightly easier for Normal errors
- Probably most sensible for experimental data?
- Examples: Ellner et al. (2002); Melbourne and Chesson (2006)

Procedure

- based on estimated observation error, pick a range of increased error values, e.g. tripling the existing observation variance in 4–8 steps
- for each error magnitude, generate a data set with that increased error (more stable to inflate a single set of errors)
- estimate parameters for each set using gradient matching (i.e. assume $\sigma^2_{\text{obs}} = 0$)
- fit a linear or quadratic regression model for parameter $= f(\text{total error})$
- extrapolate the fit to zero

Logistic fit



Kalman filter

- General approach to account for dynamic variance, expected population state
- Works for **linear** (typically Normal) models; can be extended to nonlinear models
- Natural multivariate extensions: include bias, external shocks, etc. (Schnute, 1994)

Concept and implementation

- **Concept**
  - Variance increases with process error; decreases with (accurate) observations
  - Expected population state follows expected dynamics; drawn toward (accurate) observations
- **Procedure** (pseudo-pseudo-code)
  - Run KF for specified values of parameters, σ^2_{obs}, σ^2_{proc} to compute $\hat{N}(t)$, $\sigma^2_N(t)$
  - Estimate objective function (SSQ) for $N_{\text{obs}}|\hat{N}, \sigma^2_N$
  - Minimize over $\{\text{parameters}, \sigma^2_{\text{obs}}, \sigma^2_{\text{proc}}\}$

Concept and implementation

- **Concept**
  - Variance increases with process error;
    decreases with (accurate) observations
  - Expected population state follows expected dynamics;
    drawn toward (accurate) observations
- **Procedure** (pseudo-pseudo-code)
  - Run KF for specified values of parameters, σ^2_{obs}, σ^2_{proc} to compute $\hat{N}(t)$, $\sigma^2_N(t)$
  - Estimate objective function (SSQ) for $N_{\text{obs}}|\hat{N}, \sigma^2_N$
  - Minimize over $\{\text{parameters}, \sigma^2_{\text{obs}}, \sigma^2_{\text{proc}}\}$

Autoregressive model

$$
\begin{aligned}
N(t) &\sim \text{Normal}(a + bN(t-1), \sigma^2_{\text{proc}}) \\
N_{\text{obs}}(t) &\sim \text{Normal}(N((t), \sigma^2_{\text{obs}})
\end{aligned}
$$

- $b < 1, a > 0 \rightarrow$ stable dynamics
- $b > 1 \rightarrow$ exponential growth

Procedure

1. Update mean, variance of true density according to previous expected mean and variance:

$$\begin{aligned} \text{mean}(N(t)|N_{\text{obs}}(t-1)) &\equiv \mu_1 = a + b\mu_0 \\ \text{Var}(N(t)|N_{\text{obs}}(t-1)) &\equiv \sigma_1^2 = b^2\sigma_0^2 + \sigma_{\text{proc}}^2 \end{aligned}$$

2. Now update the mean and variance of the **observed** density at time t:

$$\begin{aligned}
\text{mean}(N_{\text{obs}}(t)|N_{\text{obs}}(t-1)) &\equiv \mu_2 = \mu_1 \\
\text{Var}(N_{\text{obs}}(t)|N_{\text{obs}}(t-1)) &\equiv \sigma_2^2 = \sigma_1^2 + \sigma_{\text{obs}}^2
\end{aligned}$$

3. Now update true (expected) mean and variance to account for **current** observation:

$$\text{mean}(N|N_{\text{obs}}(t)) \equiv \mu_3 = \mu_1 + \frac{\sigma_1^2}{\sigma_2^2}(N_{\text{obs}}(t) - \mu_2)$$

$$\text{Var}(N(t)|N_{\text{obs}}(t)) \equiv \sigma_3^2 = \sigma_1^2 \left(1 - \frac{\sigma_1^2}{\sigma_2^2}\right)$$

Pseudo-code

```
KFpred <- function(params,var_proc,var_obs,init) {
  set_initial_values
  for (i in 2:nt) {
     ## ... calculate mu{1-3}, sigma^2{1-3} as above
     N[i] <- mu_3; Var[i] <- sigmasq_3
  }
  return(list(N=N,Var=Var))
}
KFobj <- function(params,var_proc,var_obs,init,Nobs)
   pred <- KFpred(params,var_proc,var_obs,init)
   obj_fun(Nobs,mean=pred$N,sd=sqrt(pred$Var))
}
minimize(KFobj,start_values,Nobs)
```

Extended Kalman filter

To fit (mildly) nonlinear models with the deterministic skeleton

$$N(t+1) = f(N(t)),$$

we just replace a and b in the autoregressive model $N(t+1) = a + bN(t)$ with the coefficients of the first two terms of the Taylor expansion of $f()$:

$$f(N(\tau)) \approx f(N(t)) + \frac{df}{dN}(N(\tau) - N(t)) + \dots$$

Multivariate extension (Schnute, 1994)

$$\begin{aligned}
\text{process: } \boldsymbol{X}_t &= \boldsymbol{A}_t + \boldsymbol{B}_t \boldsymbol{X}_{t-1} + \boldsymbol{\delta}_t \\
\text{observation: } \boldsymbol{Y}_t &= \boldsymbol{C}_t + \boldsymbol{D}_t \boldsymbol{X}_t + \boldsymbol{\epsilon}_t
\end{aligned}$$

Allows for bias, cross-species effects in both process and observation, correlation in process and observation noise . . .

Outline

State space models

- models that address the fundamental problem that the probability of a set of observations depends on the **unobserved** true values
- somehow have to deal with (integrate over?) the range of possible values of the **latent variables**
- problems are generally very high-dimensional (many unobserved values), so brute force fails: stochastic (**Monte Carlo**) integration
- exploit conditioning: if we know $N(t)$, $N(t-1)$ and $N(t)$ are **conditionally** independent

Markov chain Monte Carlo

Very general way of calculating Bayesian **posterior densities**

Gibbs sampling exploit conditioning:

$$\text{Prob}(A, B, C) \propto \text{Prob}(A|B, C) \cdot \text{Prob}(B|A, C) \cdot \text{Prob}(C|A, B)$$

This means that we can sample the conditional probabilities **sequentially** and get the right answer.

Rejection sampling (Metropolis-Hastings): we can pick new values of parameters at random, then pick a random number to decide whether to keep them. If our rule satisfies

$$\frac{\text{Prob}(A)}{\text{Prob}(B)} = \frac{P(\text{jump } B \to A) P(\text{accept } A|B)}{P(\text{jump } A \to B) P(\text{accept } B|A)}$$

then in the long run our chain will converge to the right distribution

Black boxes/magic

Given enough time and thought, you can construct your own Gibbs and Metropolis-Hastings samplers. Alternatively, you can use a powerful but opaque tool called BUGS (Bayesian Inference Using Gibbs Sampling), which exists in several incarnations (`WinBUGS`, `OpenBUGS`, `JAGS`).

BUGS allows you to specify a model in a specialized language (that looks a lot like R); it then constructs samplers for you and runs a Markov chain. It can be accessed via R (`R2jags` package) or MATLAB (`https://code.google.com/p/matbugs/`).

Black boxes/magic

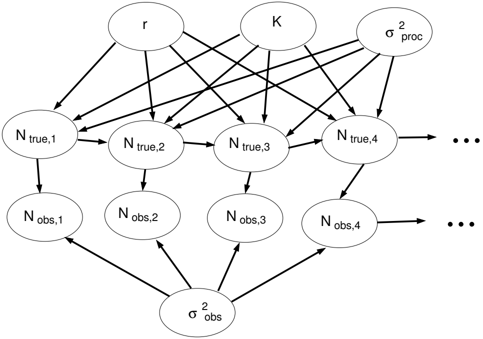
Given enough time and thought, you can construct your own Gibbs and Metropolis-Hastings samplers. Alternatively, you can use a powerful but opaque tool called BUGS (Bayesian Inference Using Gibbs Sampling), which exists in several incarnations (`WinBUGS`, `OpenBUGS`, `JAGS`).

BUGS allows you to specify a model in a specialized language (that looks a lot like R); it then constructs samplers for you and runs a Markov chain. It can be accessed via R (`R2jags` package) or MATLAB (https://code.google.com/p/matbugs/).

BUGS code for the logistic function

```
model <- function() {
  t[1] <- n0    ## initial values ...
  o[1] ~ dnorm(t[1],tau.obs)
  for (i in 2:N) {   ## step through observations ...
     v[i] <- t[i-1]+r*t[i-1]*(1-t[i-1]/K)
     t[i] ~ dnorm(v[i],tau.proc)
     o[i] ~ dnorm(t[i],tau.obs)
  }
  r ~ dunif(0.1,maxr) ## priors ...
  K ~ dgamma(0.005,0.005)
  tau.obs ~ dgamma(0.005,0.005)
  tau.proc ~ dgamma(0.005,0.005)
  n0 ~ dgamma(1,n0rate)
}
```

Dependency structure for logistic model



Running BUGS

- **Good news**: BUGS code is (relatively) intuitive
- **Bad news**:
  - Debugging is hard
  - Different parameterizations
  - Need to figure out how long to run chains (convergence diagnostics)
  - Poor mixing
  - Slow computation

Frequentist methods

- MCMC is usually done in a Bayesian framework; opens various cans of worms
- there are many other related approaches, some classical
  - sequential Monte Carlo/particle filters (Ionides et al., 2006; Doucet et al., 2001; de Valpine, 2004): R `pomp` package
  - data cloning (Lele et al., 2007): R `dclone` package

Continuous-time models

I know this is possible via particle filtering methods, but I've never tried it ...

References

de Valpine, P., 2004. *Journal of the American Statistical Association*, 99:523–536.

Doucet, A., de Freitas, N., and Gordon, N.J., 2001. *Sequential Monte Carlo methods in practice*. Springer-Verlag, New York, USA.

Ellner, S.P., Seifu, Y., and Smith, R.H., 2002. *Ecology*, 83(8):2256–2270.

Gani, R. and Leach, S., 2001. *Nature*, 414(6865):748–751.

Gillespie, D.T., 2007. *Annual Review of Physical Chemistry*, 58:35–55. ISSN 0066-426X. doi:10.1146/annurev.physchem.58.032806.104637. PMID: 17037977.

Ionides, E.L., Bretó, C., and King, A.A., 2006. *Proceedings of the National Academy of Sciences of the USA*, 103(49):18438–18443. doi:doi:10.1073pnas.0603181103.

Lele, S.R., Dennis, B., and Lutscher, F., 2007. *Ecology Letters*, 10:551–563. doi:doi:10.1111/j.1461-0248.2007.01047.x.

Melbourne, B.A. and Chesson, P., 2006. *Ecology*, 87:1478–1488.

Roughgarden, J., 1995. *Theory of Population Genetics and Evolutionary Ecology: An Introduction*. Benjamin Cummings, facsimile edition. ISBN 0134419650.

Schnute, J.T., 1994. *Canadian Journal of Fisheries and Aquatic Sciences*, 51:1676–1688.

Turelli, M., 1977. *Theoretical Population Biology*, 12(2):140–178. ISSN 00405809.

van Veen, F.J.F., van Holland, P.D., and Godfray, H.C.J., 2005. *Ecology*, 86(12):1382–1389.

Øksendal, B.K., 2003. *Stochastic differential equations: an introduction with applications*. Springer, Berlin; New York. ISBN 3540047581 9783540047582.