Estimation of parameters for stochastic dynamic models

Ben Bolker

McMaster University
Departments of Mathematics & Statistics and Biology

20 June 2017

Outline

Outline

Modeling

Typical stats	Typical math
stochastic	deterministic
static	dynamic
phenomenological	mechanistic

- Time-series models: mostly **phenomenological** and **linear** (e.g. ARIMA, spectral/wavelet analyses)
- Biomath models: mostly **mechanistic** and **nonlinear** (e.g. Lotka-Volterra, SIR, Fitzhugh-Nagumo)

Modeling

- **time**: continuous or discrete
- **state**: continuous (e.g. quantitative genetics) or discrete (e.g. Mendelian)
- **evolution**: deterministic or stochastic

e.g.

- ODEs: continuous-time, continuous-state, deterministic
- branching processes: continuous-time, discrete-state, stochastic

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

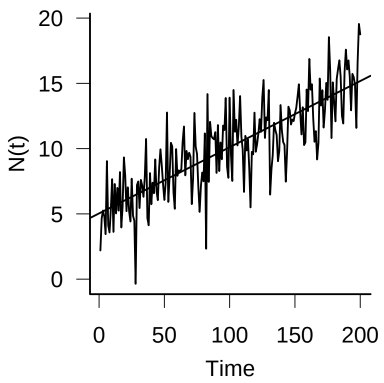
Consequences

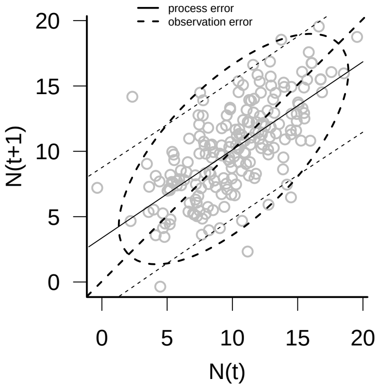
- Process error induces dynamic **changes in variance**
- Process+observation error induce **correlations** between subsequent observations
- Observation at next time step depends on **unobserved** value at current time step
- Simple statistical methods
  (i.e. uncorrelated, equal variance)
  are incorrect

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





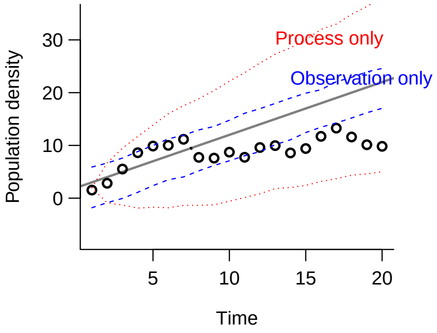
How should we interpret this single realization?

Outline

Why simulate?

- understand dynamics
- test methods in best-case scenario
- explore precision/power
- quantify properties of statistical estimators
- evaluate robustness

Linear model



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

R code (version 1)

```r
## 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)

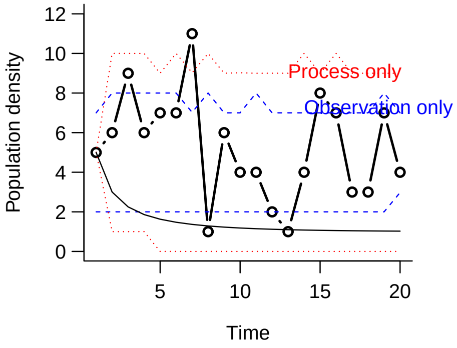
```r
library(deSolve)
linfun <- function(t,y,parms) {
    ## with() is magic to use param names directly
    g <- with(as.list(c(y,parms)), {
        N_new <- rnorm(1,mean=N+b,sd=sd_proc)
        c(N=N_new,Nobs=rnorm(1,mean=N_new,sd=sd_obs))
  })
  return(list(g))  ## deSolve needs this format
}
set.seed(101)
N0 <- c(N=a,Nobs=rnorm(1,a,sd_obs))
linparms <- c(a=6,b=1,sd_proc=sd_proc,sd_obs=sd_obs)
ode(N0,1:nt,linfun,linparms,method="iteration")
```

R code (version 3)

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

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

Hyperbolic nonlinear model



$$N(1) = N_0$$

$$N(t+1) \sim \text{Poisson}\left(\frac{aN(t)}{b+N(t)}\right)$$

$$N_{\text{obs}}(t) \sim \text{Binomial}(N(t), p)$$

R code

```
hypfun <- function(t,y,parms) {
    g <- with(as.list(c(y,parms)), {
        N_det <- a*N/(b+N)
        N <- rpois(1,lambda=N_det)
        N_obs <- rbinom(1,size=N,prob=prob_obs)
        c(N=N,Nobs=N_obs)
  })
  return(list(as.numeric(g)))  ## deSolve needs numeric() (
}
set.seed(101)
N0 <- c(N=4,N_obs=4)
hypparms <- c(a=6,b=1,prob_obs=0.9)
ode(N0,times=1:nt,func=hypfun,
    parms=hypparms,method="iteration")
```

Stochastic ODEs

- continuous-time, continuous-state
- ordinary differential equations plus dW
  (= derivative of a Brownian motion/Wiener process)
- delicate analysis (For biologists: Turelli (1977); Roughgarden (1995). For mathematicians: Øksendal (2003))
- More common for cellular/physiological than population models
- Solve via **Euler-Maruyama**
  (= Euler + appropriately scaled Gaussian noise)

(continuous-time) Markov processes

- continuous-time, discrete-state
- specify (limits of) probabilities of transitions per unit time, e.g. $P(N \to N+1)$ 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: simulate the deterministic version of the model (i.e., with neither observation nor process error) and compare
- Because measurement/observation error is (typically) independent at each observation, overall log-likelihood is sum of log-likelihood
- for Normally distributed, equal-variance error, maximum likelihood estimation equivalent to least-squares
- Very common for ODE models, e.g. Gani and Leach (2001); van Veen et al. (2005)
- Brute force can be slow/unstable: use sensitivity equations (Raue et al., 2013)

Pseudo-code

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

Real code #1 (for loops)

```r
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 #2 (using `mle2()`)

```r
library(bbmle)
obj_fun <- function(a,b,sd,nt,Nobs) {
  estimate <- determ_fun(a,b,nt)
  ## negative log-lik. of Normal
  obj <- -sum(dnorm(Nobs,estimate,sd,log=TRUE))
  return(obj)
}
determ_fun <- function(a,b,nt) a+b*(1:nt)
mle2(obj_fun,
     data=list(Nobs=linN,nt=nt),
     start=list(a=5,b=2,sd=1.01),
     method="Nelder-Mead")
```

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

Real code #3 (mle2() formula interface)

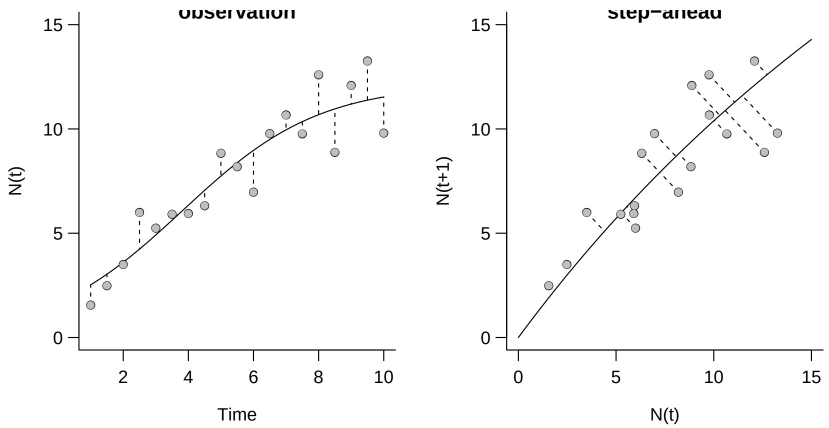
```
library(bbmle)
determ_fun <- function(a,b,nt) a+b*(1:nt)
mle2(Nobs~dnorm(determ_fun(a,b,nt),sd=sd),
     data=list(Nobs=linN,nt=nt),
     start=list(a=5,b=2,sd=1.01),
     method="Nelder-Mead")
```

Formula interface further simplifies getting predicted values, etc.
(but may make debugging harder!)

`mle2` notes

- wrapper for `optim`
- assumes objective function is negative log-likelihood
- uses `method="BFGS"` by default (maybe switch to Nelder-Mead)
- unlike `optim`, obj. function takes parameters separately: `objfun(alpha,beta)` instead of `objfun(params)`
- use `trace=TRUE` to track parameters and obj fun value
- nicer accessors (`coef()`, `logLik()`, etc.: see `methods(class="mle2")`)

Logistic model fit



Optimization tips/trouble-shooting

- use sensible starting values
  - for GA/MCMC, use values that are **different** (allow exploration) but **not crazy** (crash/get stuck)
- Nelder-Mead is slower and more robust than BFGS
- test objective/mean function externally
- use `cat()` to print parameter values, see where you're running into trouble
- use constraints (`method="L-BFGS-B"`, or BOBYQA from `nloptr` package) or transform parameters (e.g. fit $\log(\beta)$ rather than β)

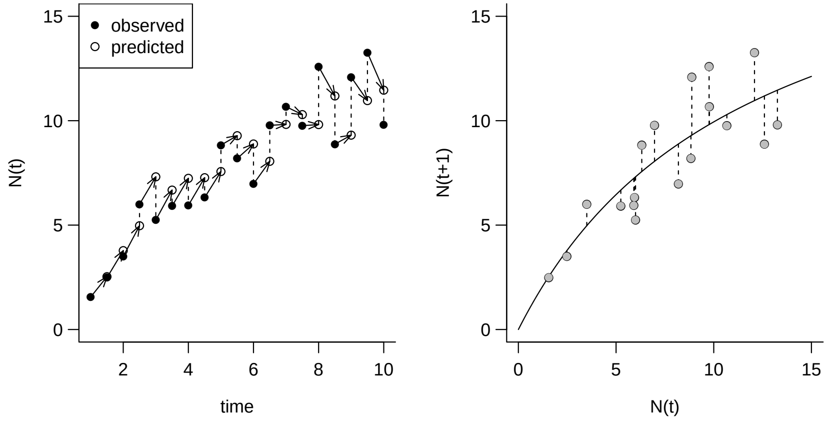
Gradient matching

- 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)
- More complicated for continuous-time models (Ellner et al., 2002): fit a smooth curve to data, then fit to derivatives of the curve

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

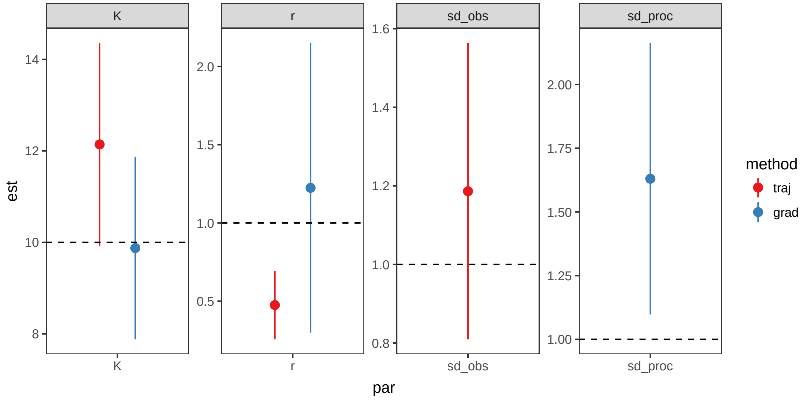


Comparison

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

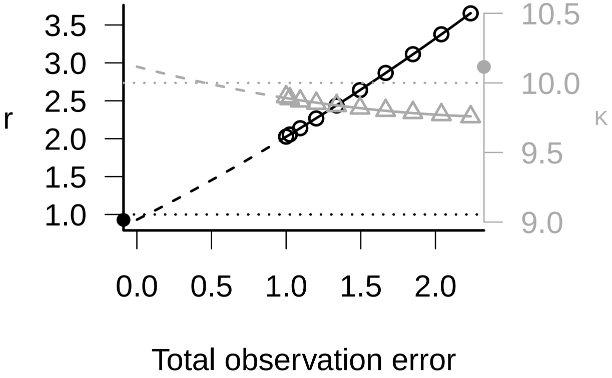
SIMEX

- **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

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

- $b < 1, a > 0 \rightarrow$ 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:

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

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

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 $f()$:

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

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 . . .

References

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.

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

Raue, A., Schilling, M., et al., 2013. *PLOS ONE*, 8(9):e74335. ISSN 1932-6203. **doi**:10.1371/journal.pone.0074335.

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.