Hierarchical Bayesian Modeling of CO2 Flux with a Matrix Exponential Solution

Introduction

See `century_experiments.pdf` for a more thorough introduction.

This documents provides results obtained by fitting a hierarchical Bayesian model of CO2 using a matrix exponential instead of an ODE integrator. We can use a matrix exponential solution for systems of *linear* ODEs, and expect such as task to be computationally less expensive. The matrix exponential solution is exact, although the computation of the matrix exponential itself is an approximation. We refer to this as a "semi-analytical" solution (which is an optimistic way of saying "semi-numerical").

R code

The R function for simulating the data is shown below. The details are described above.

```
simulate_data_century <- function(t_meas, t_cap, init_C, num_rep) {
# INPUTS:
#   t_meas:  measurement times
#   t_cap:   cap times
#   init_C:  initial pool contents
#   num_rep: number of replications
library(deSolve)
library(gtools)
genDerivs <-function(t, Ct, params) {
  # General diff eq model: dC_dt = I(t) + A(t)*C(t)
  # INPUTS:
  #   t: time
  #   Ct: the value of the vector C at time t, C(t)
  #   params: it has two fields, params$I and params$A
  dC_dt = params$I + params$A %*% Ct;
  return(list(dC_dt));
}
m <- 3; # number of pools
C_t0 <- matrix(init_C, nrow=m);
turnover <- c(1.5, 25, 1000);
K <- 1/turnover;
I <- rep(0, m); # no input flux for century
N_t <- length(t_meas);
CO2_flux_mat <- matrix(NA, nrow = N_t, ncol = num_rep);
Alpha_rep <- array(0, c(m, m, num_rep));
Alpha <- matrix(0, m, m);
# Setting global transfer rates with expert-tuned values:
Alpha[2, 1] = 0.5;
Alpha[3, 1] = 0.004;
Alpha[1, 2] = 0.42;
Alpha[3, 2] = 0.03;
Alpha[1, 3] = 0.45;
Alpha[1, 1] = 1 - Alpha[2, 1] - Alpha[3, 1];
Alpha[2, 2] = 1 - Alpha[1, 2] - Alpha[3, 2];
```

```
Alpha[3, 3] = 1 - Alpha[1, 3] - Alpha[2, 3];
for (this_rep in 1:num_rep) {
  # Hierarchical modeling of transfer rates for replications:
  kappa <- 100;
  Alpha_rep[,1, this_rep] <-  rdirichlet(1, Alpha[,1] * kappa);
  Alpha_rep[,2, this_rep] <-  rdirichlet(1, Alpha[,2] * kappa);
  Alpha_rep[,3, this_rep] <-  rdirichlet(1, Alpha[,3] * kappa);
  Alpha_rep[1, 1, this_rep] <- 0;
  Alpha_rep[2, 2, this_rep] <- 0;
  Alpha_rep[3, 3, this_rep] <- 0;
  A <- Alpha_rep[,, this_rep] * matrix(rep(K, m), nrow = m, byrow = TRUE) - diag(K);
  params <- list(I=I, A=A);
  t0 <- 0;
  # Solving the ODE system for given parameters:
  meas_data<-ode(y = C_t0, func = genDerivs,
          times = c(t0,t_meas), parms = params);
  cap_data<-ode(y = C_t0, func = genDerivs,
                times = c(t0,t_cap), parms = params);
  # Calculating CO2 flux
  totalC_t0 = sum(meas_data[1,2:(m+1)]);
  CO2_t_meas <- totalC_t0 - rowSums(meas_data[2:nrow(meas_data), 2:(m+1)]);
  CO2_t_cap <- totalC_t0 - rowSums(cap_data[2:nrow(cap_data),2:(m+1)]);
  CO2_flux <- (CO2_t_meas - CO2_t_cap)/(t_meas-t_cap);
  # Adding log-normal noise:
  CO2_flux_mat[, this_rep] <- exp(log(CO2_flux) + rnorm(length(CO2_flux),0,.5));
}
simulated_data <- list(N_t = N_t, t_meas = t_meas, t_cap = t_cap,
                       num_rep = num_rep, totalC_t0 = totalC_t0,
                       t0=t0, CO2_flux=CO2_flux_mat, Alpha_rep=Alpha_rep);
return(simulated_data);
}
```

Model specification in Stan

We study a partial pooling model (fitting a hierarchical Bayesian model which estimates parameters jointly for all replications and allows for variation between replicates.)

Functions

We redefine the function `evolved_CO2`. The ODEs are defined within the function using a matrix. The only significant edits are in the functions block. The other blocks are almost identical to those in `century_experiments.pdf`, the difference being the call to `evoloved_CO2` does not require the dummy `x_dat` and `x_int` arguments.

```
functions {
  /**
  * Compute evolved CO2 from the system given the specified
  * parameters and times. This is done by solving the century
  * model ODE system with a matrix exponential solution and
  * then calculating the rate CO2 is emmited.
  *
  * @param N_t number of times
```

```
  * @param t0 initial time
  * @param ts times
  * @param gamma partitioning coefficient
  * @param k decomposition rates
  * @param ajk transfer rates
  * @return evolved CO2 for times ts
  */
    vector evolved_CO2(int N_t, real t0, vector ts,
                       vector gamma, real totalC_t0,
                       vector k, real a21, real a31, real a12,
                       real a32, real a13) {
      vector[3] C_t0;        // initial state
      matrix[3, 3] A;        // ODE matrix
      vector[3] C_t[N_t];    // predicted pool content
      vector[N_t] CO2_t;     // evolved CO2 at times ts

      A[1, 1] = -k[1];
      A[1, 2] = a12 * k[2];
      A[1, 3] = a13 * k[3];
      A[2, 1] = a21 * k[1];
      A[2, 2] = -k[2];
      A[2, 3] = 0;
      A[3, 1] = a31 * k[1];
      A[3, 2] = a32 * k[2];
      A[3, 3] = -k[3];

      C_t0 = gamma * totalC_t0;

      for (t in 1:N_t) {
        C_t[t] = matrix_exp(ts[t] * A) * C_t0;
        CO2_t[t] = totalC_t0 - sum(C_t[t]);
      }

      return CO2_t;
    }
}
```

Fitting the models

```
num_rep <- 5;
t_meas <- c(seq(from=1/360, to=7/360, by=1/360), seq(from=14/360, to=28/360, by=7/360),
            seq(from=60/360, to=360/360, by=30/360));
t_cap <- c(seq(from=0.5/360, to=6.5/360, by=1/360), seq(from=10.5/360, to=24.5/360, by=7/360),
           seq(from=45/360, to=345/360, by=30/360));
init_C <- 1E3*c(.1, .1, .8);
data <- simulate_data_century(t_meas, t_cap, init_C, num_rep);

library(rstan);
library(tictoc);
rstan_options(auto_write = TRUE);
options(mc.cores = parallel::detectCores());
# partial pooling
```

```
tic("hier_me")
fit_hier_me <- stan("matrix_exp/century_hier_me.stan", data=data, iter=1500, seed=1234);
t4<-toc()
```

```
## hier_me: 1195.685 sec elapsed
```

hierarchical (matrix exponential)

initial carbon estimates (γ)

```
         mean se_mean   sd  2.5%   25%  50%  75% 97.5% n_eff Rhat
gamma[1] 0.15  0.0042 0.11 0.035 0.076 0.12 0.19  0.45   715    1
gamma[2] 0.50  0.0052 0.24 0.046 0.313 0.51 0.69  0.90  2142    1
gamma[3] 0.35  0.0050 0.22 0.015 0.166 0.33 0.52  0.79  2000    1
```

turnover rates

```
            mean se_mean    sd 2.5% 25%  50%  75% 97.5% n_eff Rhat
turnover[1]    2   2e-02 5e-01    1   1    2    2     3   440    1
turnover[2]   26   6e-01 2e+01   12  23   25   26    39  1570    1
turnover[3] 1304   3e+02 1e+04  543 941 1000 1059  1736  1348    1
```

transfer rates (a21)

```
       mean se_mean   sd   2.5%  25%  50%  75% 97.5% n_eff Rhat
a21[1]  0.3  0.0073 0.21 0.0098 0.12 0.27 0.46  0.74   807    1
a21[2]  0.3  0.0073 0.21 0.0112 0.13 0.27 0.46  0.73   807    1
a21[3]  0.3  0.0073 0.21 0.0107 0.12 0.27 0.46  0.74   810    1
a21[4]  0.3  0.0072 0.21 0.0089 0.12 0.27 0.45  0.74   830    1
a21[5]  0.3  0.0073 0.21 0.0092 0.13 0.27 0.46  0.74   824    1
```

transfer rates (a31)

```
       mean se_mean   sd   2.5%  25%  50%  75% 97.5% n_eff Rhat
a31[1] 0.30  0.0079 0.21 0.0103 0.12 0.25 0.45  0.75   730    1
a31[2] 0.30  0.0078 0.21 0.0106 0.12 0.26 0.46  0.75   750    1
a31[3] 0.30  0.0078 0.21 0.0100 0.12 0.26 0.46  0.75   747    1
a31[4] 0.30  0.0078 0.21 0.0099 0.12 0.26 0.45  0.75   743    1
a31[5] 0.31  0.0079 0.22 0.0107 0.13 0.26 0.47  0.76   752    1
```

Save fits for additional information