

Explanation of Contaminant1.R

Program Lines	Explanation
<pre># Script for simulating the Contaminant Cycling problem # Process Error version fitted by least squares # To run, type source("Contaminant1.R") # To clear all variables, type rm(list = ls())</pre>	<p>Comment (#) lines to explain what the program does, and how to run it</p>
<pre>rm(list = ls())</pre>	<p>Clear the memory</p>
<pre># Define rates r = c(15, 0.2, 0.45, 0.25, 0.35, 0.2, 0.15)</pre>	<p>Define the rate parameters using the concatenation operator <code>c(...)</code></p>
<pre># Organize rates into terms of difference equation # x_t+1 = F + R*x_t F = c(r[1], 0, 0) Rterms = c(1-(r[2]+r[3]+r[4]), 0, 0, r[3], 1-(r[5]+r[6]), 0, r[4], r[5], 1-r[7]) R = matrix(Rterms,nrow=3,ncol=3,byrow=TRUE)</pre>	<p>Convert the rates to the format of the difference equation. F is the vector of inputs. Rterms defines R, by rows. The last line uses the 'matrix' operator of R to convert the 9 elements of Rterms to a 3x3 matrix. <code>byrow=TRUE</code> does the conversion row by row.</p>
<pre># Define process error distribution PEmean=c(0,0,0) PEsd=c(0.3,0.3,3)</pre>	<p>Define mean and standard deviation for the process errors.</p>
<pre># Initial conditions hvore = 1 bug = 1 bird = 6 x0=c(hvore,bug,bird)</pre>	<p>Initial conditions for the simulation.</p> <p>The last line collects the initial conditions in a vector.</p>
<pre># Simulate time=0 for (i in 1:39) { x1 = F + R%%x0 + rnorm(3,mean=PEmean,sd=PEsd) x0 = x1 hvore=c(hvore,x1[1]) bug=c(bug,x1[2]) bird=c(bird,x1[3]) time=c(time,i) }</pre>	<p>Simulate the difference equation over 39 steps, from time=1 to time=40. Each step is computed between the curly brackets <code>{}</code>. The computation of <code>x1</code> uses matrix multiplication (<code>%*%</code>) and adds process error using the <code>rnorm</code> function of R (read about it in the documentation). Then <code>x0</code> is reset to <code>x1</code>, to serve as the initial value for the next time step, and values of the state variables are saved as <code>hvore</code>, <code>bug</code> and <code>bird</code> using the concatenation operator <code>c()</code>.</p>
<pre># Plot results windows() plot(time,bird,xlim=c(0,40),ylim=c(0,80),ylab="Contaminant</pre>	<p>Plot results of the simulation. <code>windows()</code> opens a new graphic window. The plot line plots <code>bird</code> vs time. The</p>

<pre>Mass", xlab="time", type='l', col='blue', lwd=2) points(time, bug, type='l', col='magenta', lwd=2) points(time, hvore, type='l', col='green', lwd=2) title(main="Bird-blue, Bug-magenta, Hvore-green") grid()</pre>	<p>'points' lines add additional variables to the same graphic window. The 'title' line adds a title, and grid() overlays a grid. See the documentation for more information about plot, point, and title.</p>
<pre># Estimate parameters by Least Squares process error fit</pre>	<p>Now we will treat the simulation results as "data" and try to recover the parameter values</p>
<pre># Define function for sum of squared errors SSE <- function(p){ F1 = c(p[1], 0, 0) Rterms1 = c(1-(p[2]+p[3]+p[4]), 0, 0, p[3], 1-(p[5]+p[6]), 0, p[4], p[5], 1-p[7]) R1 = matrix(Rterms1, nrow=3, ncol=3, byrow=TRUE) err=0 for (i in 1:39) { x0 = c(hvore[i], bug[i], bird[i]) y = c(hvore[i+1], bug[i+1], bird[i+1]) yhat = F1 + R1%*%x0 err = c(err, (y-yhat)) } sumerr2 = sum(err*err) return(sumerr2) }</pre>	<p>This function computes the sum of squared errors given a set of parameter values in the vector p. First the inputs F1 and rate matrix R1 are calculated from the parameter vector.</p> <p>Then a variable 'err' is initialized to hold the errors. The 'for' loop runs over 39 steps, between the curly braces. At each step, x0 is the data at the beginning of the step. y is then computed from x0, using F1 and R. Errors, y-yhat, are then collected in the 'err' vector using the concatenation operator.</p> <p>After the 'for' loop is complete, err is squared and summed, and the sum of squared errors is returned.</p>
<pre># Check error with true parameters SSEtrue <- SSE(r)</pre>	<p>These lines compute the sum of squared errors for the true parameters</p>
<pre># Form initial guesses of parameters guess = r*rnorm(7, mean=1, sd=0.5)</pre>	<p>These lines create an initial guess of the parameters, as random perturbations of the true parameters using the R function rnorm.</p>
<pre># Error with guess SSEguess <- SSE(guess)</pre>	<p>These lines compute the sum of squared errors for the parameter guesses</p>
<pre># Nonlinear Model Fit; try 'nlm' or 'optim' ModFit <- nlm(SSE, p=guess, print.level=1) rest <- ModFit\$estimate minSSE <- ModFit\$minimum</pre>	<p>These lines fit the model using the R function 'nlm', saves the results as ModFit, and extracts the parameter estimates as 'rest' and the minimum sum of squared errors as 'minSSE'.</p>
<pre># Print output print(c("SSE with True parameters", SSEtrue), quote=FALSE) print(c("SSE with Guessed</pre>	<p>These lines print useful output. The argument quote=FALSE suppresses printing of quote marks. The</p>

<pre>parameters", SSEguess), quote=FALSE) print(c("Minimum SSE", minSSE), quote=FALSE) print("True Parameters", quote=FALSE) print(r) print("Estimated Parameters", quote=FALSE) print(round(rest, 2))</pre>	<p>'round' command rounds off 'rest', in this case to 2 figures after the decimal.</p>
<pre># Calculate predictions p <- rest # Substitute estimated parameters into p Fp = c(p[1], 0, 0) Rtermsp = c(1-(p[2]+p[3]+p[4]), 0, 0, p[3], 1-(p[5]+p[6]), 0, p[4], p[5], 1-p[7]) Rp = matrix(Rtermsp, nrow=3, ncol=3, byrow=TRUE) yhat1=hvore[1] yhat2=bug[1] yhat3=bird[1] for (i in 1:39) { x0 = c(hvore[i], bug[i], bird[i]) x1 = Fp + Rp%*%x0 yhat1 = c(yhat1, x1[1]) yhat2 = c(yhat2, x1[2]) yhat3 = c(yhat3, x1[3]) }</pre>	<p>These lines compute predictions using the estimated parameters in 'rest'. Note the similarity to the original simulation lines above. The re-use of the same logic suggests that it might have been more efficient to program this segment as a function.</p>
<pre># Plot predictions and observations windows() plot(yhat3, bird, xlim=c(0, 80), ylim=c(0, 80), ylab=" ", xlab=" ", type='p', pch=24, col='blue') title(main="Bird-blue, Bug-magenta, Hvore- green", ylab="Observation", xlab="Prediction") points(yhat2, bug, type='p', pch=25, col='magenta') points(yhat1, hvore, type='p', pch=19, col='green') points(c(0, 80), c(0, 80), type='l', col='black', lwd=1) grid()</pre>	<p>These lines plot predictions and observations. The last call to 'points' plots a straight line through the origin with slope = 1, as a benchmark for accuracy of the predictions.</p>

R functions to look up in the manual or html help:

matrix

function

concatenation operator c()

nlm

windows()

plot

points

title

grid

print

round

for

rnorm