Supplementary Materials File S1. Generation of Synthetic Data

Below is the MATLAB code implementing the hybrid method for generating synthetic data. Kim and Park's basic method was combined with Cichocki et al.'s idea of using orthogonal components. To induce correlations in the synthetic data, the Cholesky decomposition of the desired correlation matrix is provided as an input matrix U. The definition of the function StepFunctionColumns is shown on the next page.

```
function [ simData ] = CorrelatedSyntheticData1(m,n,k,sparsity,noiseMag,U)
% m        : Number of rows.
% n        : Number of columns.
% k        : Number of components.
% sparsity : Sparsity level (e.g., 0.4 for 40% sparsity)
% noiseMag : magnitude of the noise (e.g., 0.05 for 5% of the average magnitude
%            of elements in simulated data
% U        : Cholesky decomposition of the Correlation Matrix C; U = chol(C);

% we randomly constructed kx matrix H with 40% sparsity. (Park and Kim)
H                  = rand(k,n); % Uniformly distributed in [0,1]
keepH              = rand(k,n); % Uniformly distributed in [0,1]
H(keepH<=sparsity) = 0.0;

% Set W to a series of orthogonal step functions.
% (Following Cichocki et al.'s "signal processing" approach.)
W = StepFunctionColumns(m,k);

% Apply the correlations to the data to the W matrix.
W = W * U;

% Compute A (Park and Kim, and also Cichocki et al., construct A in this way).
A = W * H;

% and added Gaussian noise to each element where the standard
% deviation is 5% of the average magnitude of elements in A.  (Park and Kim)
avgMagA = mean(A(:));              % Compute average of elements of A.
mag     = noiseMag * avgMagA;      % 5% of the average magnitude of elements in A
A       = A + ( mag * randn(m,n)); % add Gaussian noise
simData = abs(A);                     % Make sure that A is non-negative after adding
    Gaussian noise.
function W = StepFunctionColumns(nRows, nCols)
%
% This function generates a matrix whose rows are simple orthogonal step functions
%
% INPUTS:
% nRows  - desired number of rows
% nCols  - desired number of columns
%
% OUTPUT:
%
% W - matrix whose rows are simple orthogonal step functions

% If nRows < nCols, exit out with error.
if ( nRows < nCols )
    error('StepFunctionRows: nRows must be greater than or equal to nCols!')
end

W = zeros(nRows,nCols);

% Loop over columns, design step functions
stepFactor = nRows / nCols ;
startIndexTmp = 1; % Initialize startIndex.
for i=1:nCols
    % Compute endIndex from startIndex.
    endIndexTmp = startIndexTmp + stepFactor - 1;
    startIndex  = round(startIndexTmp);
```

```
    endIndex    = round(endIndexTmp);

    % If this is the last column, make sure the 1's should go all the way to the last
row.
    if ( i == nCols)
        endIndex = nRows;
    end

    % Set select elements in the i-th column of W to 1.
    W(startIndex:endIndex,i) = ones(endIndex-startIndex+1,1);

    % Update startIndex.
    startIndexTmp = endIndexTmp + 1;
end
```