Supplement of

Technical Note: Partial wavelet coherency for improved understanding of scale-specific and localized bivariate relationships in geosciences

Wei Hu and Bing Si

Correspondence to: Wei Hu (wei.hu@plantandfood.co.nz)

Introduction

S1 Matlab code for calculating multiple wavelet coherence

S2 Matlab code for significance test on multiple wavelet coherence

S3 User manual for S2 (mwc.m) and S3 (mwcsignif.m)

S4 Artificial data used for method test

S5 Results of wavelet power spectrum and bivariate wavelet coherency for real datasets

S1 Matlab code for MWC (mwc.m)

```
% This is a Matlab code (pwc.m) for calculating partial wavelet coherency.
% Please copy the following content into a txt file and rename it to "pwc.m"
% prior to running.

function varargout=pwc(X,varargin)
%  Partial Wavelet coherency
% Creates a figure of partial wavelet coherency
% USAGE: [Rsq,period,scale,coi,sig95]=pwc(X,[,settings])
%
% Input: X: a matrix of multiple variables equally distributed in space
%             or time. The first column corresponds to the dependent variable,
%          the second column corresponds to the independent variable,
%         and the third and subsequent columns are excluding variables.
%
% Settings: Pad: pad the time series with zeros?
% .         Dj: Octaves per scale (default: '1/12')
% .         S0: Minimum scale
% .         J1: Total number of scales
% .         Mother: Mother wavelet (default 'morlet')
% .         MaxScale: An easier way of specifying J1
% .         MakeFigure: Make a figure or simply return the output.
% .         BlackandWhite: Create black and white figures
% .         AR1: the ar1 coefficients of the series
% .              (default='auto' using a naive ar1 estimator. See ar1nv.m)
% .         MonteCarloCount: Number of surrogate data sets in the
%             significance calculation. (default=300)

% Settings can also be specified using abbreviations. e.g. ms=MaxScale.
% For detailed help on some parameters type help wavelet.
% Example:
%    t=[1:200]';
%    pwc([sin(t),sin(t.*cos(t*.01)),cos(t.*sin(t*.01))]) % for one
%    excluding variable
%    pwc([sin(t),sin(t.*cos(t*.01)),cos(t),cos(t.*sin(t*.01))])  % for two
%    excluding variables
% Please acknowledge the use of this software package in any publications,
% by including text such as:

%   "The software for the partial wavelet coherency was provided by Wei Hu
%    and is available in the Supplement of Hu and Si(2020)
% (http://to be determined)."
%   and cite the paper:
% "Hu, W., and Si,B (2020), Technical Note: Partial wavelet coherency for
% improved understanding of scale-specific and localized
% bivariate relationships in geosciences, Hydrol. Earth Syst. Sci., to
% be determined."
%
%  (C) W. Hu 2020
%
% ------------------------------------------------------------
%
%   Copyright (C) 2020, W. Hu 2020
%   This software may be used, copied, or redistributed as long as it is not
%   sold and this copyright notice is reproduced on each copy made.  This
%   routine is provided as is without any express or implied warranties
%   whatsoever.
%
%    Wavelet software was provided by C. Torrence and G. Compo,
%       and is available at URL: http://paos.colorado.edu/research/wavelets/.
%
```

```
%    Crosswavelet and wavelet coherence software were provided by
%      Aslak Grinsted and is available at URL:
%    http://www.glaciology.net/wavelet-coherence
%
%        Multiple wavelet coherence software are provided by
%           Hu and Si (2016) and is available at URL:
%      (https://www.hydrol-earth-syst-sci.net/20/3183/2016/hess-20-3183-2016-
%     supplement.pdf)

%        Partial wavelet coherency software are provided by
%           Hu and Si (2020) and is available at URL:
%          (to be determined)
%   We acknowledge Aslak Grinsted for his wavelet coherency code (wtc.m) on
% which this code builds.
%
%---------parse function arguments-----------------------------------------

[row,col]=size(X);
[y1,dt1]=formatts(X(:,1));
mm=y1(1,1);
nn=y1(end,1);

[y2,dt2]=formatts(X(:,2));
mm2=y2(1,1);
nn2=y2(end,1);

for i=3:col;
[x,dtx]=formatts(X(:,i));

if (dt1~=dtx |dt2~=dtx)
    error('timestep must be equal between time series');
end

mm1=x(1,1);
nn1=x(end,1);


mm=max([mm,mm1,mm2]);
nn=min([nn,nn1,nn2]);

x1(:,(i-2))=x(:,1);
x2(:,(i-2))=x(:,2);

end

t=(mm:dt1:nn)';


%common time period
if length(t)<4
    error('The three time series must overlap.');
end

n=length(t);

%----------default arguments for the wavelet transform-----------
Args=struct('Pad',1,...      % pad the time series with zeroes (recommended)
            'Dj',1/12, ...    % this will do 12 sub-octaves per octave
            'S0',2*dt1,...    % this says start at a scale of 2 years
            'J1',[],...
            'Mother','Morlet', ...
            'MaxScale',[],...   %a more simple way to specify J1
            'MakeFigure',(nargout==0),...
```

```
            'MonteCarloCount',300,...
            'BlackandWhite',0,...
            'AR1','auto',...
            'ArrowDensity',[30 30],...
            'ArrowSize',1,...
            'ArrowHeadSize',1);

Args=parseArgs(varargin,Args,{'BlackandWhite'});

if isempty(Args.J1)
    if isempty(Args.MaxScale)
        Args.MaxScale=(n*.17)*2*dt1; %auto maxscale;
    end
    Args.J1=round(log2(Args.MaxScale/Args.S0)/Args.Dj);
end

ad=mean(Args.ArrowDensity);
Args.ArrowSize=Args.ArrowSize*30*.03/ad;
%Args.ArrowHeadSize=Args.ArrowHeadSize*Args.ArrowSize*220;
Args.ArrowHeadSize=Args.ArrowHeadSize*120/ad;

if ~strcmpi(Args.Mother,'morlet')
    warning('MWC:InappropriateSmoothingOperator','Smoothing operator is
designed for morlet wavelet.');
end

if strcmpi(Args.AR1,'auto')
      for i=1:col
arc(i)= ar1nv(X(:,i));
       end
Args.AR1=arc
    if any(isnan(Args.AR1))
        error('Automatic AR1 estimation failed. Specify it manually (use arcov
or arburg).');
    end
end

%----------::::::::::::::---------- ANALYZE ----------::::::::::::-----------

%Calculate and smooth wavelet spectrum Y1, Y2, and X

[Y1,period,scale,coiy1] =
wavelet(y1(:,2),dt1,Args.Pad,Args.Dj,Args.S0,Args.J1,Args.Mother);
sinv=1./(scale');
smY1=smoothwavelet(sinv(:,ones(1,n)).*(abs(Y1).^2),dt1,period,Args.Dj,scale);
dte=dt1*.01;
idx=find((y1(:,1)>=(t(1)-dte))&(y1(:,1)<=(t(end)+dte)));
Y1=Y1(:,idx);
smY1=smY1(:,idx);
coiy1=coiy1(idx);
coi1=coiy1;


[Y2,period,scale,coiy2] =
wavelet(y2(:,2),dt1,Args.Pad,Args.Dj,Args.S0,Args.J1,Args.Mother);
sinv=1./(scale');
smY2=smoothwavelet(sinv(:,ones(1,n)).*(abs(Y2).^2),dt1,period,Args.Dj,scale);
dte=dt2*.01;
idx=find((y2(:,1)>=(t(1)-dte))&(y2(:,1)<=(t(end)+dte)));
Y2=Y2(:,idx);
smY2=smY2(:,idx);
coiy2=coiy2(idx);
coi2=coiy2;
```

```
for  i=3:col
 [XS,period,scale,coix] = wavelet(x2(:,(i-
2)),dt1,Args.Pad,Args.Dj,Args.S0,Args.J1,Args.Mother);

idx=find((x1(:,(i-2))>=(t(1))-dte)&(x1(:,(i-2))<=(t(end)+dte)));
XS=XS(:,idx);
coix=coix(idx);

XS1(:,:,(i-2))=XS;
coi0=min(coi1,coi2);
coi=min(coi0,coix);
end

% -------- Calculate Cross Wavelet Spectra----------------------------

% -- between dependent variable (or independent variables) and excluding
% factors------

for i=1:(col-2)
Wy1x=Y1.*conj(XS1(:,:,i));
sWy1x=smoothwavelet(sinv(:,ones(1,n)).*Wy1x,dt1,period,Args.Dj,scale);
sWy1x1(:,:,i)=sWy1x;

Wy2x=Y2.*conj(XS1(:,:,i));
sWy2x=smoothwavelet(sinv(:,ones(1,n)).*Wy2x,dt1,period,Args.Dj,scale);
sWy2x1(:,:,i)=sWy2x;
end

% ---- between dependent variable and independent variables------
Wy1y2=Y1.*conj(Y2);
sWy1y2=smoothwavelet(sinv(:,ones(1,n)).*Wy1y2,dt1,period,Args.Dj,scale);

% ----between excluding variables and excluding variables------
for i=1:(col-2);
for j=1:(col-2);
Wxx=XS1(:,:,i).*conj(XS1(:,:,j));
sWxx=smoothwavelet(sinv(:,ones(1,n)).*Wxx,dt1,period,Args.Dj,scale);
sWxx1(:,:,i,j)=sWxx;
end
end

% --------------- Partial wavelet coherence --------------------------------
% calculate the partial wavelet coherence
m=length(scale);
Rsq=zeros(m,n);
Wy1y2x=zeros(m,n);
nofe=col-2; %number of excluding factors

if nofe==1;
% ------Partial wavelet coherence for one excluding variable ------------------
%%%%%%%%% |1-R^2 y,x·Z (s,t)|^2   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for ii=1:m;
    for jj=1:n;
        sWxx1_i(ii,jj)=inv(sWxx1(ii,jj)); % inversed sWxx1
    end
end

sWy1x1_sWxx1=bsxfun(@times,sWy1x1,sWxx1_i); % sWy1x1*sWxx1_i
sWy2x1_c=conj(sWy2x1); %conjugate of smoothed cross-wavelet power spectra
% between independent variable and excluding variables
```

```
sWy1x1_sWxx1_sWy2x1=bsxfun(@times,sWy1x1_sWxx1,sWy2x1_c);
% sWy1x1*sWxx1_i*sWy2x1_c
R2y1y2x=sWy1x1_sWxx1_sWy2x1./sWy1y2; %(Ry,x.Z)^2
abs_one_minus_R2y1y2x=(abs(1-R2y1y2x)).^2; %squared absolute of 1-(Ry,x.Z)^2
%%%%%%%%%%%%%%%% R^2 y,x (s,t)%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
R2y1y2=(sWy1y2.*conj(sWy1y2))./(smY1.*smY2); %(Ry,x)^2
%%%%%%%%%%%%%%%%%%% 1-R^2 y,Z(s,t)%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
sWy1x1_c=conj(sWy1x1);
sWy1x1_sWxx1_sWy1x1=bsxfun(@times,sWy1x1_sWxx1,sWy1x1_c);
R2y1x=sWy1x1_sWxx1_sWy1x1./smY1; %(Ry,Z)^2
one_minus_R2y1x=(1-R2y1x);

%%%%%%%%%%%%%%%%%%%%%%  1-R^2 x,Z(s,t) %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
sWy2x1_sWxx1=bsxfun(@times,sWy2x1,sWxx1_i);
sWy2x1_sWxx1_sWy2x1=bsxfun(@times,sWy2x1_sWxx1,sWy2x1_c);
R2y2x=sWy2x1_sWxx1_sWy2x1./smY2; %(Rx,Z)^2
one_minus_R2y2x=(1-R2y2x);
%%%%%%%%%%%%%%%%%%%  R^2 y,x·Z %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Rsq_num=roundn(bsxfun(@times,abs_one_minus_R2y1y2x,R2y1y2),-16);
%numerator part of the equation for squared pwc, if the value is less than
%10^(-16), 0 is assigned
Rsq_den=real(bsxfun(@times,one_minus_R2y1x,one_minus_R2y2x));
%denominator part of the equation for squared pwc
Rsq= bsxfun(@rdivide,Rsq_num,Rsq_den); %squared partial wavelet coherence
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

Wy1y2x=1-R2y1y2x;

else

% -----Partial wavelet coherence for two or more excluding variables ----------

%%%%%%%% |1-R^2 y,x·Z (s,t)|^2   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
sWy1x1_p=permute(sWy1x1,[3,1,2]); %re-structure sWy1x1
sWy1x1_rp=reshape(sWy1x1_p,[1,nofe,m,n]);%change 3D to 4D matrix
sWxx1_p=permute(sWxx1,[4,3,1,2]); % re-structure sWxx1 (smoothed auto-or cross-
% wavelet power spectra for excluding factors)
for jj=1:n;
    for ii=1:m;
        sWxx1_ip(:,:,ii,jj)=inv(sWxx1_p(:,:,ii,jj)); % inversed sWxx1
    end
end
sWy1x1_sWxx1=bsxfun(@times,sWy1x1_rp,sWxx1_ip); % sWy1x1_rp*sWxx1_ip
sWy1x1_sWxx1_s=sum(sWy1x1_sWxx1,2);
sWy1x1_sWxx1_ps=permute(sWy1x1_sWxx1_s,[2,1,3,4]);  % re-structure
sWy1x1_sWxx1_sps=squeeze(sWy1x1_sWxx1_ps); % 4D to 3D

sWy2x1_c=conj(sWy2x1); %conjugate of smoothed cross-wavelet power spectra
% between independent variable and excluding variables
sWy2x1_pc=permute(sWy2x1_c,[3,1,2]); % re-structure

sWy1x1_sWxx1_sWy2x1=bsxfun(@times,sWy1x1_sWxx1_sps,sWy2x1_pc);
% sWy1x1_rp*sWxx1_ip*sWy2x1_pc
sWy1x1_sWxx1_sWy2x1_s=sum(sWy1x1_sWxx1_sWy2x1,1);
sWy1x1_sWxx1_sWy2x1_ss=squeeze(sWy1x1_sWxx1_sWy2x1_s);

R2y1y2x=sWy1x1_sWxx1_sWy2x1_ss./sWy1y2; %(Ry,x.Z)^2
abs_one_minus_R2y1y2x=(abs(1-R2y1y2x)).^2; %squared absolute of 1-(Ry,x.Z)^2
%%%%%%%%%%%%%%%% R^2 y,x (s,t)%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
R2y1y2=(sWy1y2.*conj(sWy1y2))./(smY1.*smY2); %(Ry,x)^2
%%%%%%%%%%%%%%%%%%% 1-R^2 y,Z(s,t)%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
sWy1x1_c=conj(sWy1x1);
sWy1x1_pc=permute(sWy1x1_c,[3,1,2]);
```

```
sWy1x1_sWxx1_sWy1x1=bsxfun(@times,sWy1x1_sWxx1_sps,sWy1x1_pc);
sWy1x1_sWxx1_sWy1x1_s=sum(sWy1x1_sWxx1_sWy1x1,1);
sWy1x1_sWxx1_sWy1x1_ss=squeeze(sWy1x1_sWxx1_sWy1x1_s);
R2y1x=sWy1x1_sWxx1_sWy1x1_ss./smY1; %(Ry,Z)^2
one_minus_R2y1x=(1-R2y1x);

%%%%%%%%%%%%%%%%%%%%%%  1-R^2 x,Z(s,t) %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
sWy2x1_p=permute(sWy2x1,[3,1,2]);
sWy2x1_rp=reshape(sWy2x1_p,[1,nofe,m,n]);
sWy2x1_sWxx1=bsxfun(@times,sWy2x1_rp,sWxx1_ip);
sWy2x1_sWxx1_s=sum(sWy2x1_sWxx1,2);
sWy2x1_sWxx1_ps=permute(sWy2x1_sWxx1_s,[2,1,3,4]);
sWy2x1_sWxx1_sps=squeeze(sWy2x1_sWxx1_ps);
sWy2x1_sWxx1_sWy2x1=bsxfun(@times,sWy2x1_sWxx1_sps,sWy2x1_pc);
sWy2x1_sWxx1_sWy2x1_s=sum(sWy2x1_sWxx1_sWy2x1,1);
sWy2x1_sWxx1_sWy2x1_ss=squeeze(sWy2x1_sWxx1_sWy2x1_s);
R2y2x=sWy2x1_sWxx1_sWy2x1_ss./smY2; %(Rx,Z)^2
one_minus_R2y2x=(1-R2y2x);
%%%%%%%%%%%%%%%%%%%  R^2 y,x·Z %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Rsq_num=roundn(bsxfun(@times,abs_one_minus_R2y1y2x,R2y1y2),-16);
%numerator part of the equation for squared pwc, if the value is less than
% 10^(-16), 0 is assigned
Rsq_den=real(bsxfun(@times,one_minus_R2y1x,one_minus_R2y2x));
%denominator part of the equation for squared pwc
Rsq= bsxfun(@rdivide,Rsq_num,Rsq_den); %squared partial wavelet coherence
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

Wy1y2x=1-R2y1y2x;
end


% --------------- make figure-----------------------------------------------
if (nargout>0)||(Args.MakeFigure)

pwcsig=pwcsignif(Args.MonteCarloCount,Args.AR1,dt1,length(t)*2,Args.Pad,Args.Dj
,Args.S0,Args.J1,Args.Mother,.6);
    pwcsig=(pwcsig(:,2))*(ones(1,n));
    pwcsig=Rsq./pwcsig;
end

if Args.MakeFigure

    Yticks = 2.^(fix(log2(min(period))):fix(log2(max(period))));

    if Args.BlackandWhite
        levels = [0 0.5 0.7 0.8 0.9 1];
        [cout,H]=safecontourf(t,log2(period),Rsq,levels);

        colorbarf(cout,H)
        cmap=[0 1;.5 .9;.8 .8;.9 .6;1 .5];
        cmap=interp1(cmap(:,1),cmap(:,2),(0:.1:1)');
        cmap=cmap(:,[1 1 1]);
        colormap(cmap)
        set(gca,'YLim',log2([min(period),max(period)]), ...
            'YDir','reverse', 'layer','top', ...
            'YTick',log2(Yticks(:)), ...
            'YTickLabel',num2str(Yticks'), ...
            'layer','top');
        ylabel('Period');
        hold on

        %phase plot
       aWxy=angle(Wy1y2)+angle(Wy1y2x);
```

```
                aaa=aWxy;
        aaa(Rsq<.5)=NaN; %remove phase indication where Rsq is low
           aaa(isnan(Rsq))=NaN; %remove phase indication where Rsq is NaN
           %[xx,yy]=meshgrid(t(1:5:end),log2(period));

        phs_dt=round(length(t)/Args.ArrowDensity(1));
tidx=max(floor(phs_dt/2),1):phs_dt:length(t);
        phs_dp=round(length(period)/Args.ArrowDensity(2));
pidx=max(floor(phs_dp/2),1):phs_dp:length(period);

phaseplot(t(tidx),log2(period(pidx)),aaa(pidx,tidx),Args.ArrowSize,Args.ArrowHe
adSize);

        if ~all(isnan(pwcsig))
            [c,h] = contour(t,log2(period),pwcsig,[1 1],'k');%#ok
            set(h,'linewidth',2);
        end
        %suptitle([sTitle ' coherence']);
        %plot(t,log2(coi),'k','linewidth',2)
                tt=[t([1 1])-dt1*.5;t;t([end end])+dt1*.5];
        %hcoi=fill(tt,log2([period([end 1]) coi period([1 end])]));
        %hatching- modified by Ng and Kwok
        hcoi=fill(tt,log2([period([end 1]) coi period([1 end])]),'w');

        hatch(hcoi,45,[0 0 0]);
        hatch(hcoi,135,[0 0 0]);
        set(hcoi,'alphadatamapping','direct','facealpha',.5);
        plot(t,log2(coi),'color','black','linewidth',1.5);
        hold off
    else
        H=imagesc(t,log2(period),Rsq);%#ok
        %[c,H]=safecontourf(t,log2(period),Rsq,[0:.05:1]);
        %set(H,'linestyle','none')

        set(gca,'clim',[0 1]);

        HCB=safecolorbar;%#ok

        set(gca,'YLim',log2([min(period),max(period)]), ...
            'YDir','reverse', 'layer','top', ...
            'YTick',log2(Yticks(:)), ...
            'YTickLabel',num2str(Yticks'), ...
            'layer','top');
        ylabel('Period');
        hold on

  %phase plot
       aWxy=angle(Wy1y2)+angle(Wy1y2x);
                aaa=aWxy;
        aaa(Rsq<.5)=NaN; %remove phase indication where Rsq is low
        aaa(isnan(Rsq))=NaN; %remove phase indication where Rsq is NaN
        %[xx,yy]=meshgrid(t(1:5:end),log2(period));

        phs_dt=round(length(t)/Args.ArrowDensity(1));
tidx=max(floor(phs_dt/2),1):phs_dt:length(t);
        phs_dp=round(length(period)/Args.ArrowDensity(2));
pidx=max(floor(phs_dp/2),1):phs_dp:length(period);

phaseplot(t(tidx),log2(period(pidx)),aaa(pidx,tidx),Args.ArrowSize,Args.ArrowHe
adSize);

        if ~all(isnan(pwcsig))
            [c,h] = contour(t,log2(period),pwcsig,[1 1],'k');%#ok
```

```
            set(h,'linewidth',2);
        end
        %suptitle([sTitle ' coherence']);
        tt=[t([1 1])-dt1*.5;t;t([end end])+dt1*.5];
        hcoi=fill(tt,log2([period([end 1]) coi period([1 end])]),'w');
        set(hcoi,'alphadatamapping','direct','facealpha',.5);
        hold off
    end
end
%-----------------------------------------------------------------%

varargout={Rsq,period,scale,coi,pwcsig};
varargout=varargout(1:nargout);

function [cout,H]=safecontourf(varargin)
vv=sscanf(version,'%i.');
if (version('-release')<14)|(vv(1)<7)
    [cout,H]=contourf(varargin{:});
else
    %[cout,H]=contourf('v6',varargin{:});
    [cout,H]=contourf(varargin{:});
end

function hcb=safecolorbar(varargin)
vv=sscanf(version,'%i.');

if (version('-release')<14)|(vv(1)<7)
    hcb=colorbar(varargin{:});
else
    %hcb=colorbar('v6',varargin{:});
    hcb=colorbar(varargin{:});
end
```

S2 Matlab code for significance test on partial wavelet coherency

```
% This is a Matlab file (pwcsignif.m) for calculating significance tests on
%  partial wavelet coherency.
%Please copy the following content into a txt file and rename this file to
% "pwcsignif.m" prior to running.

function pwcsig=pwcsignif(mccount,ar1,dt,n,pad,dj,s0,j1,mother,cutoff)
% Partial Wavelet Coherence Significance Calculation (Monte Carlo)
%
% pwcsig=pwcsignif(mccount,ar1,dt,n,pad,dj,s0,j1,mother,cutoff)
%
% mccount: number of time series generations in the monte carlo run
%(the greater the better)
% ar1: a vector of AR1 coefficients.
% dt,pad,dj,s0,j1,mother: see wavelet help...
% n: length of each generated timeseries. (obsolete)
%
% cutoff: (obsolete)
%
% RETURNED
% pwcsig: the 95% significance level as a function of scale...
% (scale,sig95level)
% ------------------------------------------------------------
% Please acknowledge the use of this software package in any publications,
% by including text such as:
%
%   "The software for the partial wavelet coherence was provided by Wei Hu
%    and is available in the Supplement of Hu and Si (2020)
% (http://to be determined)."
```

```
%   and cite the paper:
% "Hu, W. and Si, B (2020), Technical Note: Partial wavelet coherency for
% improved understanding of scale-specific and localized
% bivariate relationships in geosciences, Hydrol. Earth Syst. Sci., to
% be determined."
%
%  (C) W. Hu 2020
%  Copyright (C) 2020, W. Hu 2020
% ------------------------------------------------------------
%   This software may be used, copied, or redistributed as long as it is not
%   sold and this copyright notice is reproduced on each copy made.  This
%   routine is provided as is without any express or implied warranties
%   whatsoever.
%
%    Wavelet software was provided by C. Torrence and G. Compo,
%       and is available at URL: http://paos.colorado.edu/research/wavelets/.
%
%    Crosswavelet and wavelet coherence software were provided by
%      Aslak Grinsted and is available at URL:
%    http://www.glaciology.net/wavelet-coherence
%
%        Multiple wavelet coherence software are provided by
%           Hu and Si (2016) and is available at URL:
%         (https://www.hydrol-earth-syst-sci.net/20/3183/2016/hess-20-3183-
%       2016-supplement.pdf)

%        Partial wavelet coherence software are provided by
%           Hu and Si (2020) and is available at URL:
%         (to be determined)

% We acknowledge Aslak Grinsted for his code (wtcsignif.m) on
% which this code builds.
%
%---------------------------------------------------------------------
cachedir=fileparts(mfilename('fullpath'));
cachedir=fullfile(cachedir,'.cache');

%we don't need to do the monte carlo if we have a cached
%siglevel for ar1s that are almost the same. (see fig4 in Grinsted et al.,
2004)
aa=round(atanh(ar1(:)')*4); %this function increases the sensitivity near 1 & -
1
aa=abs(aa)+.5*(aa<0); %only positive numbers are allowed in the checkvalues
(because of log)

% do a check that it is not the same as last time... (for optimization
purposes)
checkvalues=single([aa dj s0/dt j1 double(mother)]); %n & pad are not
important.
%also the resolution is not important.

checkhash=['' mod(sum(log(checkvalues+1)*127),25)+'a'
mod(sum(log(checkvalues+1)*54321),25)+'a'];

cachefilename=fullfile(cachedir,['pwcsignif-cached-' checkhash '.mat']);

%the hash is used to distinguish cache files.
try
    last=load(cachefilename);
    if (last.mccount>=mccount) && (isequal(checkvalues,last.checkvalues))
        pwcsig=last.pwcsig;
        return
    end
```

```
catch
end

%choose a n so that largest scale have atleast some part outside the coi
ms=s0*(2^(j1*dj))/dt; %maxscale in units of samples
n=ceil(ms*6);

warned=0;
%precalculate stuff that's constant outside the loop
%d1=ar1noise(n,1,ar1(1),1);
d1=rednoise(n,ar1(1),1);
[W1,period,scale,coi] = wavelet(d1,dt,pad,dj,s0,j1,mother);
outsidecoi=zeros(size(W1));
for s=1:length(scale)
    outsidecoi(s,:)=(period(s)<=coi);
end
sinv=1./(scale');
sinv=sinv(:,ones(1,size(W1,2)));

if mccount<1
    mwcsig=scale';
    mwcsig(:,2)=.71; %pretty good
    return
end

sig95=zeros(size(scale));

maxscale=1;
for s=1:length(scale)
    if any(outsidecoi(s,:)>0)
        maxscale=s;
    else
        sig95(s)=NaN;
        if ~warned
warning('Long wavelengths completely influenced by COI. (suggestion: set n
higher, or j1 lower)');
         warned=1;
        end
    end
end


nbins=1000;
wlc=zeros(length(scale),nbins);

wbh = waitbar(0,['Running Monte Carlo (significance)... (H=' checkhash
')'],'Name','Monte Carlo (PWC)');

for ii=1:mccount
    waitbar(ii/mccount,wbh);

dy1=rednoise(n,ar1(1),1);
[Wdy1,period,scale,coiy] = wavelet(dy1,dt,pad,dj,s0,j1,mother);
sinv=1./(scale');
smdY1=smoothwavelet(sinv(:,ones(1,n)).*(abs(Wdy1).^2),dt,period,dj,scale);


dy2=rednoise(n,ar1(2),1);
[Wdy2,period,scale,coiy] = wavelet(dy2,dt,pad,dj,s0,j1,mother);
sinv=1./(scale');
smdY2=smoothwavelet(sinv(:,ones(1,n)).*(abs(Wdy2).^2),dt,period,dj,scale);

col=size(ar1,2);
```

```
for  i=3:col
dx=rednoise(n,ar1(i),1);
 [Wdx,period,scale,coix] = wavelet(dx,dt,pad,dj,s0,j1,mother);
Wdx1(:,:,(i-2))=Wdx;
end

% -------- Calculate Cross Wavelet Spectra----------------------------


% -between dependent variable (or independent variables) and excluding factors-

for i=1:(col-2)
Wdy1x=Wdy1.*conj(Wdx1(:,:,i));
sWdy1x=smoothwavelet(sinv(:,ones(1,n)).*Wdy1x,dt,period, dj,scale);
sWdy1x1(:,:,i)=sWdy1x;

Wdy2x=Wdy2.*conj(Wdx1(:,:,i));
sWdy2x=smoothwavelet(sinv(:,ones(1,n)).*Wdy2x,dt,period, dj,scale);
sWdy2x1(:,:,i)=sWdy2x;
end

% ---- between dependent variable and independent variables------
Wdy1y2=Wdy1.*conj(Wdy2);
sWdy1y2=smoothwavelet(sinv(:,ones(1,n)).*Wdy1y2,dt,period,dj,scale);

% ----between excluding variables and excluding variables-
for i=1:(col-2);
for j=1:(col-2);
Wdxx=Wdx1(:,:,i).*conj(Wdx1(:,:,j));
sWdxx=smoothwavelet(sinv(:,ones(1,n)).*Wdxx,dt,period,dj,scale);
sWdxx1(:,:,i,j)=sWdxx;
end
end

% --------------- Partial wavelet coherence ---------------------------------
% calculate the partial wavelet coherence
m=length(scale);
Rsq=zeros(m,n);
nofe=col-2; %number of excluding factors

if nofe==1;
% --------Partial wavelet coherency for one excluding variable ----------------
%%%%%%%%%%%% |1-R^2 y,x·Z (s,t)|^2   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for ii=1:m;
    for jj=1:n;
        sWdxx1_i(ii,jj)=inv(sWdxx1(ii,jj)); % inversed sWxx1
    end
end

sWdy1x1_sWdxx1=bsxfun(@times,sWdy1x1,sWdxx1_i); % sWy1x1*sWxx1_i
sWdy2x1_c=conj(sWdy2x1); %conjugate of smoothed cross-wavelet power spectra
% between independent variable and excluding variables
sWdy1x1_sWdxx1_sWdy2x1=bsxfun(@times,sWdy1x1_sWdxx1,sWdy2x1_c);
% sWy1x1*sWxx1_i*sWy2x1_c
R2y1y2x=sWdy1x1_sWdxx1_sWdy2x1./sWdy1y2; %(Ry,x.Z)^2
abs_one_minus_R2y1y2x=(abs(1-R2y1y2x)).^2; %squared absolute of 1-(Ry,x·Z)^2
%%%%%%%%%%%%%%%% R^2 y,x (s,t)%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
R2y1y2=(sWdy1y2.*conj(sWdy1y2))./(smdY1.*smdY2); %(Ry,x)^2
%%%%%%%%%%%%%%%%%%% 1-R^2 y,Z(s,t)%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
sWdy1x1_c=conj(sWdy1x1);
sWdy1x1_sWdxx1_sWdy1x1=bsxfun(@times,sWdy1x1_sWdxx1,sWdy1x1_c);
R2y1x=sWdy1x1_sWdxx1_sWdy1x1./smdY1; %(Ry,Z)^2
```

```
one_minus_R2y1x=(1-R2y1x);

%%%%%%%%%%%%%%%%%%%%%%  1-R^2 x,Z(s,t) %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
sWdy2x1_sWdxx1=bsxfun(@times,sWdy2x1,sWdxx1_i);
sWdy2x1_sWdxx1_sWdy2x1=bsxfun(@times,sWdy2x1_sWdxx1,sWdy2x1_c);
R2y2x=sWdy2x1_sWdxx1_sWdy2x1./smdY2; %(Rx,Z)^2
one_minus_R2y2x=(1-R2y2x);
%%%%%%%%%%%%%%%%%%%%  R^2 y,x·Z %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Rsq_num=roundn(bsxfun(@times,abs_one_minus_R2y1y2x,R2y1y2),-16);
%numerator part of the equation for squared pwc, if the value is less than
% 10^(-16), 0 is assigned
Rsq_den=real(bsxfun(@times,one_minus_R2y1x,one_minus_R2y2x));
%denominator part of the equation for squared pwc
Rsq= bsxfun(@rdivide,Rsq_num,Rsq_den); %squared partial wavelet coherence
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

else

% ------- Partial wavelet coherence for two or more excluding variables -------

%%%%%%%%%%%%%%%|1-R^2 y,x·Z (s,t)|^2   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
sWdy1x1_p=permute(sWdy1x1,[3,1,2]); %re-structure sWy1x1
sWdy1x1_rp=reshape(sWdy1x1_p,[1,nofe,m,n]);%change 3D to 4D matrix
sWdxx1_p=permute(sWdxx1,[4,3,1,2]); % re-structure sWxx1 (smoothed auto-or
% cross-wavelet power spectra for excluding factors)
for jj=1:n;
    for ii=1:m;
        sWdxx1_ip(:,:,ii,jj)=inv(sWdxx1_p(:,:,ii,jj)); % inversed sWxx1
    end
end
sWdy1x1_sWdxx1=bsxfun(@times,sWdy1x1_rp,sWdxx1_ip); % sWy1x1_rp*sWxx1_ip
sWdy1x1_sWdxx1_s=sum(sWdy1x1_sWdxx1,2);
sWdy1x1_sWdxx1_ps=permute(sWdy1x1_sWdxx1_s,[2,1,3,4]);  % re-structure
sWdy1x1_sWdxx1_sps=squeeze(sWdy1x1_sWdxx1_ps); % 4D to 3D

sWdy2x1_c=conj(sWdy2x1); %conjugate of smoothed cross-wavelet power spectra
% between independent variable and excluding variables
sWdy2x1_pc=permute(sWdy2x1_c,[3,1,2]); % re-structure

sWdy1x1_sWdxx1_sWdy2x1=bsxfun(@times,sWdy1x1_sWdxx1_sps,sWdy2x1_pc);
% sWy1x1_rp*sWxx1_ip*sWy2x1_pc
sWdy1x1_sWdxx1_sWdy2x1_s=sum(sWdy1x1_sWdxx1_sWdy2x1,1);
sWdy1x1_sWdxx1_sWdy2x1_ss=squeeze(sWdy1x1_sWdxx1_sWdy2x1_s);

R2y1y2x=sWdy1x1_sWdxx1_sWdy2x1_ss./sWdy1y2; %(Ry,x.Z)^2
abs_one_minus_R2y1y2x=(abs(1-R2y1y2x)).^2; %squared absolute of 1-(Ry,x.Z)^2
%%%%%%%%%%%%%%%% R^2 y,x (s,t)%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
R2y1y2=(sWdy1y2.*conj(sWdy1y2))./(smdY1.*smdY2); %(Ry,x)^2
%%%%%%%%%%%%%%%%%%%% 1-R^2 y,Z(s,t)%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
sWdy1x1_c=conj(sWdy1x1);
sWdy1x1_pc=permute(sWdy1x1_c,[3,1,2]);
sWdy1x1_sWdxx1_sWdy1x1=bsxfun(@times,sWdy1x1_sWdxx1_sps,sWdy1x1_pc);
sWdy1x1_sWdxx1_sWdy1x1_s=sum(sWdy1x1_sWdxx1_sWdy1x1,1);
sWdy1x1_sWdxx1_sWdy1x1_ss=squeeze(sWdy1x1_sWdxx1_sWdy1x1_s);
R2y1x=sWdy1x1_sWdxx1_sWdy1x1_ss./smdY1; %(Ry,Z)^2
one_minus_R2y1x=(1-R2y1x);

%%%%%%%%%%%%%%%%%%%%%%  1-R^2 x,Z(s,t) %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
sWdy2x1_p=permute(sWdy2x1,[3,1,2]);
sWdy2x1_rp=reshape(sWdy2x1_p,[1,nofe,m,n]);
sWdy2x1_sWdxx1=bsxfun(@times,sWdy2x1_rp,sWdxx1_ip);
sWdy2x1_sWdxx1_s=sum(sWdy2x1_sWdxx1,2);
sWdy2x1_sWdxx1_ps=permute(sWdy2x1_sWdxx1_s,[2,1,3,4]);
```

```
sWdy2x1_sWdxx1_sps=squeeze(sWdy2x1_sWdxx1_ps);
sWdy2x1_sWdxx1_sWdy2x1=bsxfun(@times,sWdy2x1_sWdxx1_sps,sWdy2x1_pc);
sWdy2x1_sWdxx1_sWdy2x1_s=sum(sWdy2x1_sWdxx1_sWdy2x1,1);
sWdy2x1_sWdxx1_sWdy2x1_ss=squeeze(sWdy2x1_sWdxx1_sWdy2x1_s);
R2y2x=sWdy2x1_sWdxx1_sWdy2x1_ss./smdY2; %(Rx,Z)^2
one_minus_R2y2x=(1-R2y2x);
%%%%%%%%%%%%%%%%%%%  R^2 y,x·Z %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Rsq_num=roundn(bsxfun(@times,abs_one_minus_R2y1y2x,R2y1y2),-16);
%numerator part of the equation for squared pwc, if the value is less than
% 10^(-16), 0 is assigned
Rsq_den=real(bsxfun(@times,one_minus_R2y1x,one_minus_R2y2x));
%denominator part of the equation for squared pwc
Rsq= bsxfun(@rdivide,Rsq_num,Rsq_den); %squared partial wavelet coherence
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
end

 for s=1:maxscale
        cd=Rsq(s,find(outsidecoi(s,:)));
        cd=max(min(cd,1),0);
        cd=floor(cd*(nbins-1))+1;
        for jj=1:length(cd)
            wlc(s,cd(jj))=wlc(s,cd(jj))+1;
        end
    end
end
close(wbh);

for s=1:maxscale
    rsqy=((1:nbins)-.5)/nbins;
    ptile=wlc(s,:);
    idx=find(ptile~=0);
    ptile=ptile(idx);
    rsqy=rsqy(idx);
    ptile=cumsum(ptile);
    ptile=(ptile-.5)/ptile(end);
    sig95(s)=interp1(ptile,rsqy,.95);
end
pwcsig=[scale' sig95'];

if any(isnan(sig95))&(~warned)
    warning('Sig95 calculation failed. (Some NaNs)');
else
    try
        save(cachefilename,'mccount','checkvalues','pwcsig');
%save to a cache....
    catch
        warning(['Unable to write to cache file: ' cachefilename]);
    end
end
```

S3 User manual for S1 (pwc.m) and S2 (pwcsignif.m)

Partial wavelet coherency package
by Wei Hu

Release date: 28 April 2020

This software package is written for performing partial wavelet coherency.

This software package includes pwc.m and pwcsignif.m, which are written in the Matlab program based on wtc.m and wtcsignif.m provided by Aslak Grinsted (http://www.glaciology.net/wavelet-coherence) and mwc.m and mwcsignif.m provided by Wei Hu and Bing Si (https://www.hydrol-earth-syst-sci.net/20/3183/2016/hess-20-3183-2016-supplement.pdf).

Users are, therefore, required to download Aslak Grinsted's software package and combine these two packages into one to run the partial wavelet coherency analysis.

Please acknowledge the use of this software package in any publications by including text such as:

****

The software for the partial wavelet coherency was provided by Wei Hu, and is available in the supplement of Hu and Si (2020) (http://*to be determined*).

****

and cite the paper:

%%

Hu, W., and Si, B (2020), Technical Note: Partial wavelet coherency for improved understanding of scale-specific and localized bivariate relationships in geosciences, Hydrol. Earth Syst. Sci., *to be determined*.

%%

Acknowledgements:

Wavelet software is provided by C. Torrence and G. Compo,
and is available at URL: http://paos.colorado.edu/research/wavelets/.

Crosswavelet and wavelet coherence software are provided by
Aslak Grinsted and are available at URL:
http://www.glaciology.net/wavelet-coherence

Multiple wavelet coherence software is provided by Wei Hu and Bing Si and is available at:
https://www.hydrol-earth-syst-sci.net/20/3183/2016/hess-20-3183-2016-supplement.pdf
Should there be any enquiries, please feel free to contact:

Wei Hu
Email: wei.hu@plantandfood.co.nz

S4 Artificial data used for method test

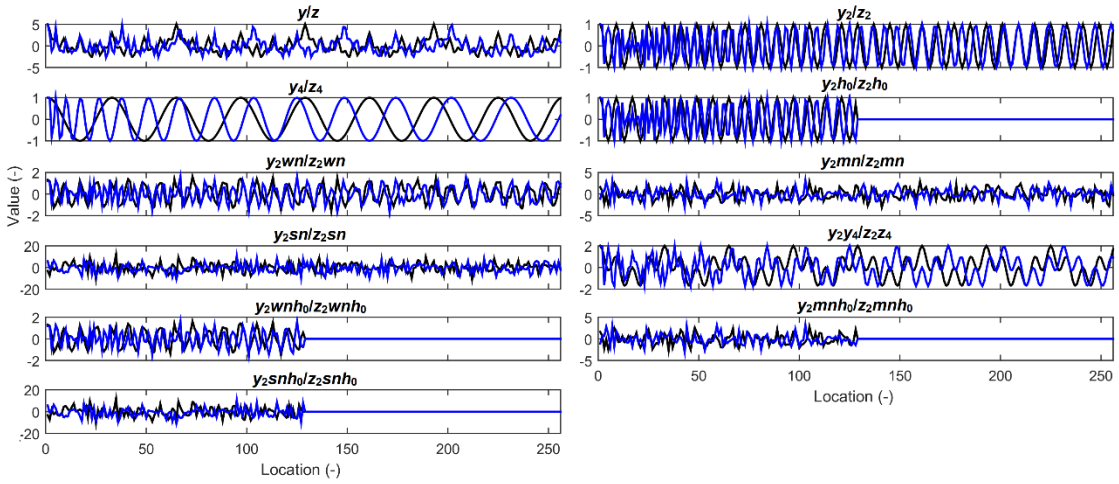

Figure S1. Variables used to test partial wavelet coherency. Black and blue lines represent variables for stationary (i.e., y, y_2, y_4, y_2h_0, y_2wn, y_2mn, y_2sn, y_2y_4, y_2wnh_0, y_2mnh_0, and y_2snh_0) and non-stationary (e.g., z, z_2, z_4, z_2h_0, z_2wn, z_2mn, z_2sn, z_2z_4, z_2wnh_0, z_2mnh_0, and z_2snh_0) cases, respectively. All variables are explained in section 3.1 of the main body of the paper.

S5 Results of wavelet power spectrum and bivariate wavelet coherency for real datasets

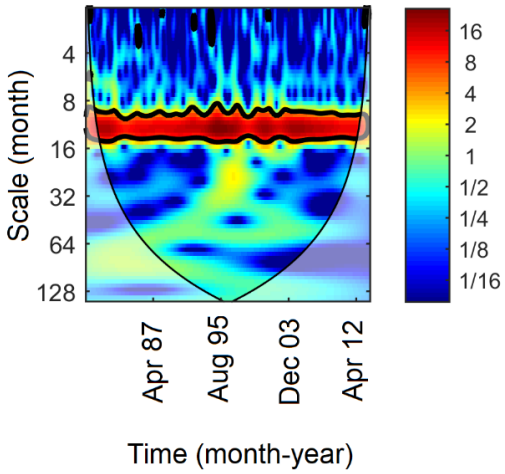

Figure S2. Wavelet power spectrum of free water evaporation (E) at the Changwu site in Shaanxi, China. Thin solid lines demarcate the cones of influence, and thick solid lines show the 95% confidence levels.

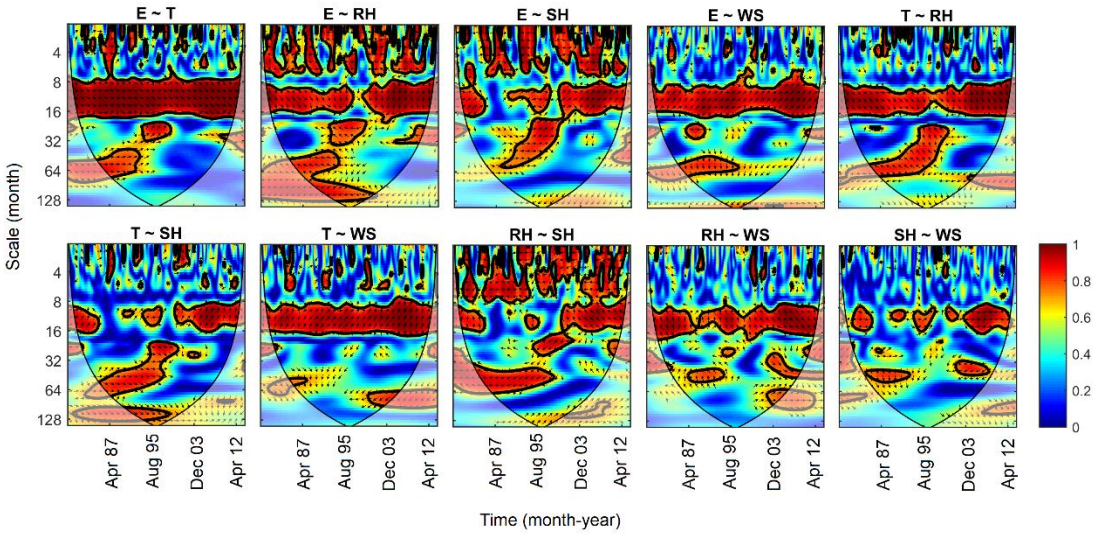

Figure S3. Bivariate wavelet coherency between every two meteorological factors (E, evaporation; T, mean temperature; RH, relative humidity; SH, sun hours; WS, wind speed) at the Changwu site in Shaanxi, China. Arrows show the phase angles of the wavelet spectra. Thin solid lines demarcate the cones of influence, and thick solid lines show the 95% confidence levels.

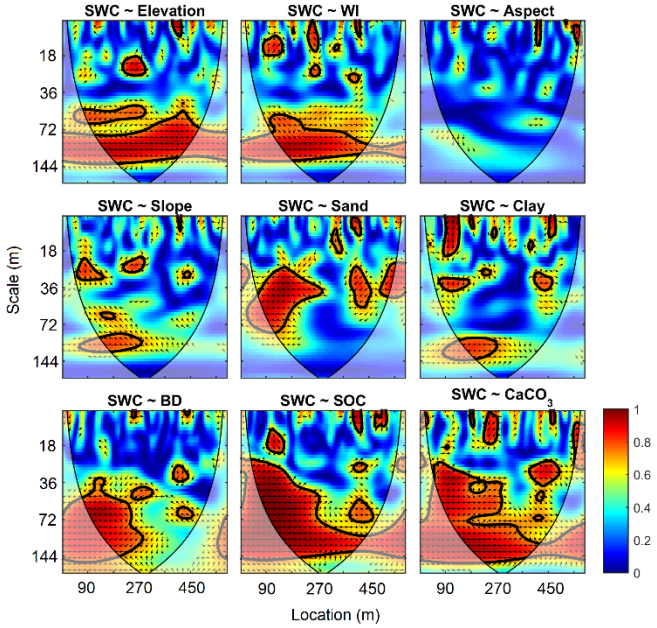

Figure S4. Bivariate wavelet coherency between soil water content (SWC) in spring and each of environmental factors (SOC, soil organic carbon; $CaCO_3$, depth to the $CaCO_3$ layer; WI, wetness index; BD, bulk density) in the transect of the hummocky landscape of the Canadian Prairies. The horizontal axis is the sampling location compared with the transect origin. Arrows show the phase angles of the wavelet spectra. Thin solid lines demarcate the cones of influence, and thick solid lines show the 95% confidence levels.

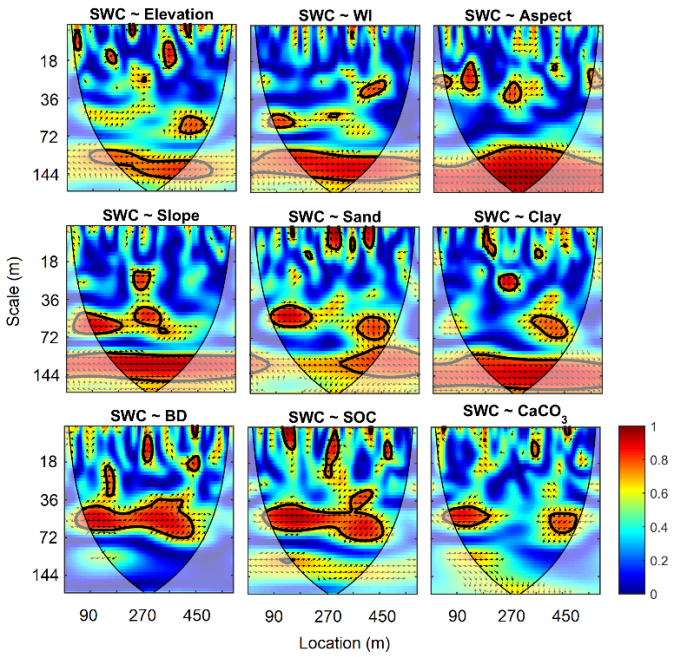

Figure S5. Bivariate wavelet coherency between soil water content (SWC) in summer and each of environmental factors (Aspect, Cos(aspect); SOC, soil organic carbon; $CaCO_3$, depth to the $CaCO_3$ layer; WI, wetness index; BD, bulk density) in the transect of the hummocky landscape of the Canadian Prairies. The horizontal axis is the sampling location compared with the transect origin. Arrows show the phase angles of the wavelet spectra. Thin solid lines demarcate the cones of influence, and thick solid lines show the 95% confidence levels.

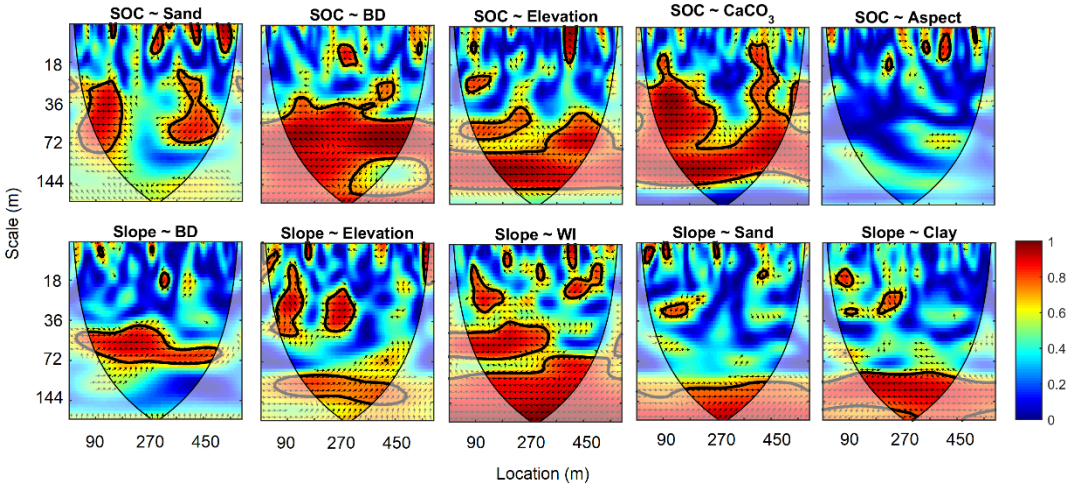

Figure S6. Bivariate wavelet coherency between two environmental factors (Aspect, Cos(aspect); SOC, soil organic carbon; $CaCO_3$, depth to the $CaCO_3$ layer; WI, wetness index; BD, bulk density) in the transect of the hummocky landscape of the Canadian Prairies. The horizontal axis is the sampling location compared with the transect origin. Arrows show the phase angles of the wavelet spectra. Thin solid lines demarcate the cones of influence, and thick solid lines show the 95% confidence levels.

References

Grinsted, A., Moore, J. C., and Jevrejeva, S.: Application of the cross wavelet transform and wavelet coherence to geophysical time series, Nonlinear Proc. Geoph., 11, 561–566, 2004.

Hu, W., and Si, B.C.: Technical note: Multiple wavelet coherence for untangling scale-specific and localized multivariate relationships in geosciences, Hydrol. Earth Syst. Sc., 20, 3183–3191, 2016.