Supplement of Nat. Hazards Earth Syst. Sci., 19, 1087–1103, 2019
https://doi.org/10.5194/nhess-19-1087-2019-supplement

Natural Hazards and Earth System Sciences
Open Access
EGU

Supplement of

Chilling accumulation in fruit trees in Spain under climate change

Alfredo Rodríguez et al.

Correspondence to: Alfredo Rodríguez (alfre2ky@gmail.com, alfredo.rodriguez@uclm.es)

Table S1. CMIP5/CORDEX EUR-11 bias-adjusted simulation matrix. CORDEX EUR-11 RCMs (Giorgi and Gutowski, 2015) and CMIP5 GCMs (Taylor et al., 2012) used (all r1i1p1/v1 ensemble/downscaling). Grey cells indicate the GCM-RCM combinations selected. The Institute ID of each model is provided in bold italics. Daily maximum and minimum temperatures were retrieved from the Earth System Grid Federation (ESGF) servers.

Bias-corrected *CMIP5/CORDEX EUR-11* *GCM/RCMs*	**CNRM-CM5** *CNRM-France*	**EC-EARTH** *ECMWF-European*	**IPSL-CM5A-MR** *IPSL-France*	**HadGEM2-ES** *Met Office-UK*	**MPI-ESM-LR** *MPI-Germany*
CCLM4-8-17 *CCLM community-International*					
REMO2009 *MPI-Germany*					
RCA4 *SMHI-Sweden*					
RACMO22E *KNMI-Netherlands*					
ALADIN53 *CHMI-Czech Republic*					
WRF331F *IPSL-France*					

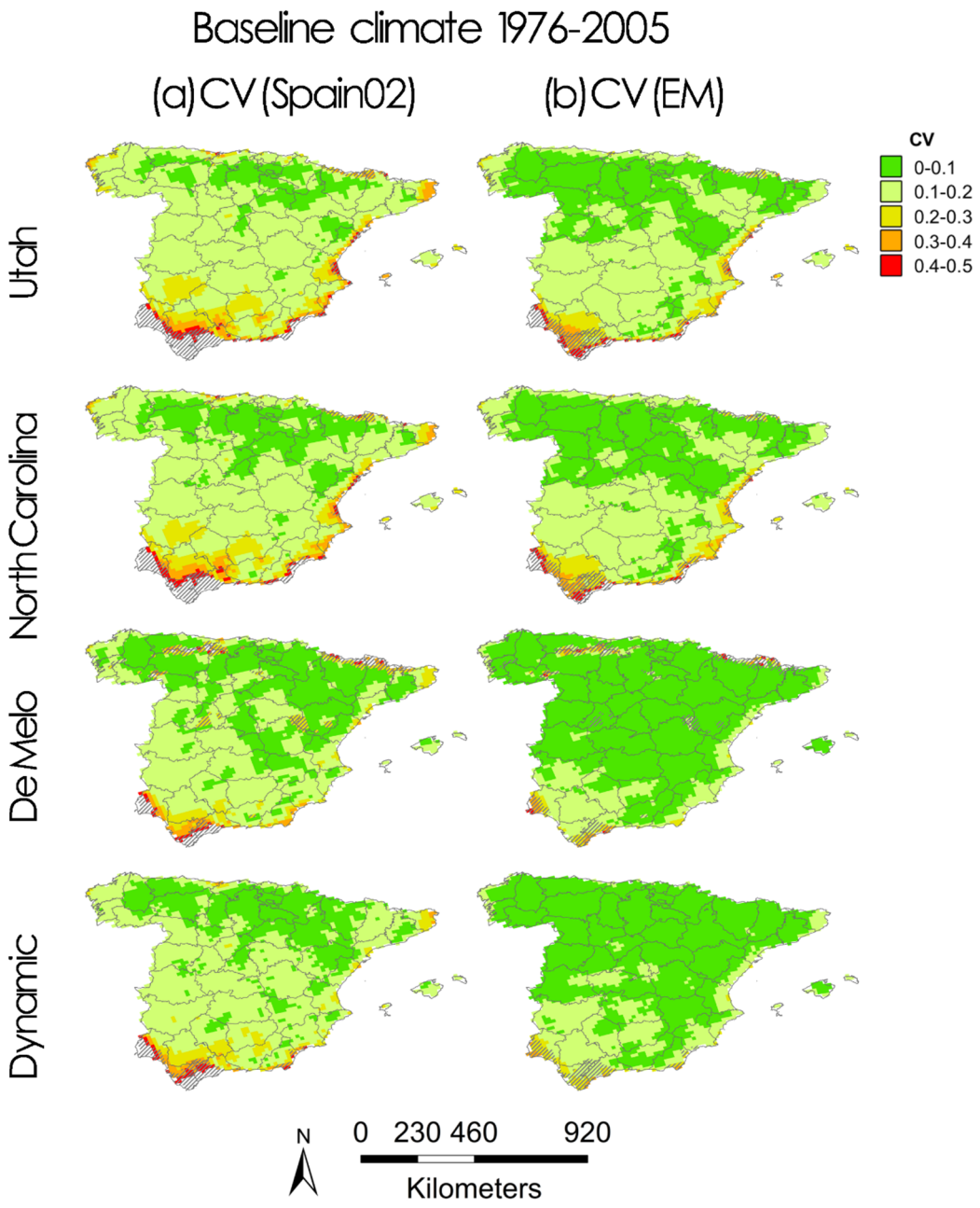

Figure S1. Comparison of chilling accumulation inter-annual variability for the baseline period (1975–2005). For each chilling method (rows), the coefficient of variation (CV, expressed per unit) of the chilling accumulation calculated with the observational data set Spain02 (first column) and with the 10 EUR-11 ensemble members' mean CV of the 30-year period (second column). Grid cells with mean absolute percentage error (MAPE, %) values from the validation phase higher than 20 are not considered as reliable results and are highlighted in diagonal lines.

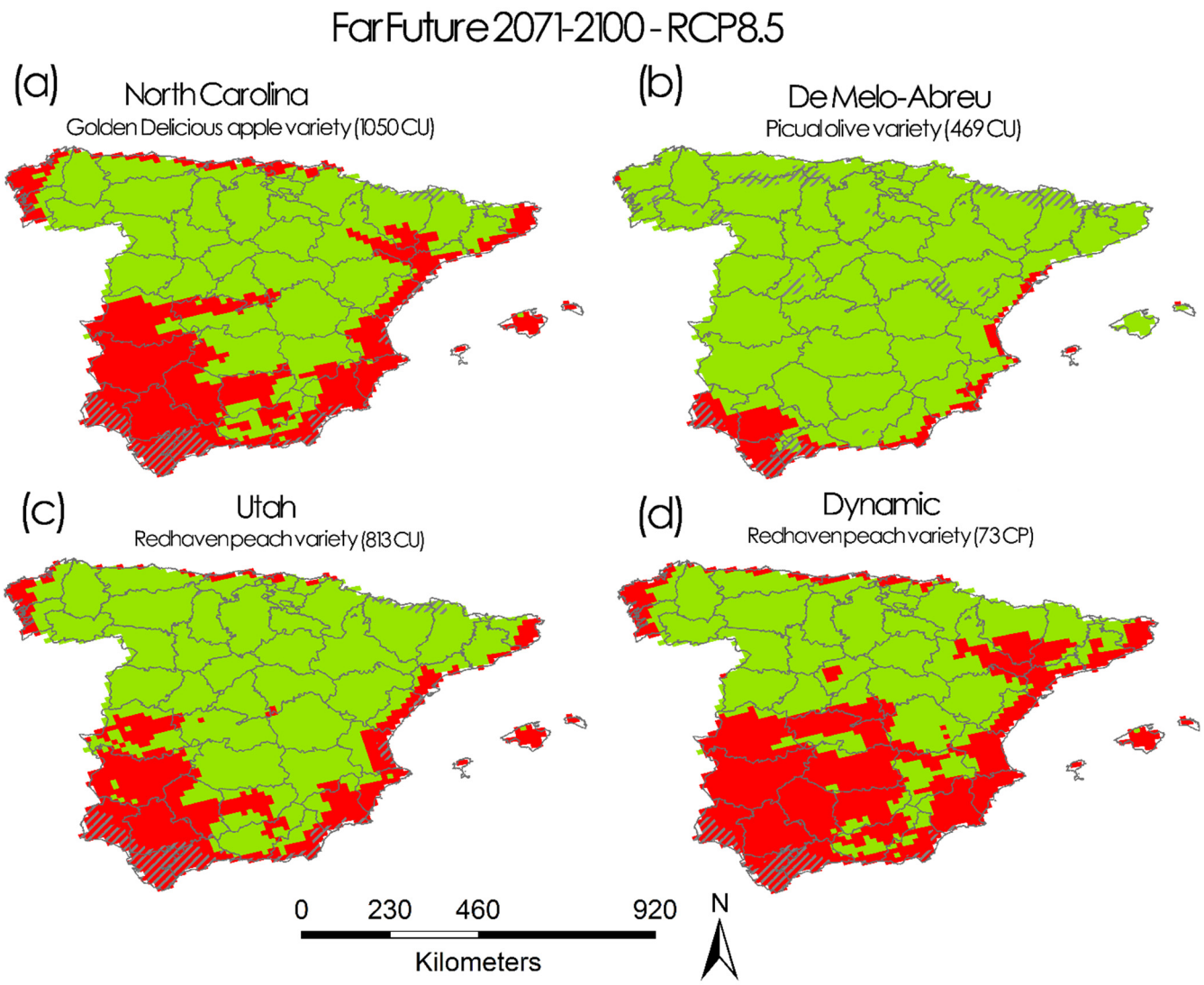

Figure S2. Map of areas with chilling accumulation, in the 2071-2100 period and for RCP8.5 scenario, projected to be higher (green grid cells) or lower (red grid cells) than 1050 chilling units (chilling requirements estimated for Golden Delicious apple variety with the North Carolina model, plot a), than 469 chilling units (chilling requirements estimated for Picual olive variety with the De Melo-Abreu model, plot b), than 813 chilling units (chilling requirements estimated for Redhaven peach variety with the Utah model, plot c) or than 73 chilling portions (chilling requirements estimated for Redhaven peach variety with the Dynamic model, plot d). Grid cells with mean absolute percentage error (MAPE, %) values from the validation phase higher than 20 are not considered as reliable results and are highlighted in diagonal lines.

Code S1. Function *chill_portions* MATLAB code, for calculating the chilling portions accumulated by the Dynamic model (Fishman et al., 1987a; Fishman et al., 1987b), with the standard parameters and the equations following Luedeling and Brown (2011)

```
function [delt] = chill_portions(t,time)

% Julian day to cut between seasons
DIA_CORTE = 186;

% See the "cuts" now in hourly approach
pos = [1];
for i=min(time(:,1)):max(time(:,1))
   x = time(:,1) == i;
   aux = find(x);
   aux = aux((DIA_CORTE-1)*24+1:end);
   if ~isempty(aux)
       pos = [pos aux(1)];
   end
end

% Constants
e0 = 4153.5;
e1 = 12888.8;
a0 = 139500;
a1 = 2567000000000000000;
slp = 1.6;
tetmlt = 277;
aa = a0/a1;
ee = e1-e0;

tk = t+273.15;
ftmprt = slp*tetmlt*(tk-tetmlt)./tk;
sr = exp(ftmprt);

xi = sr./(1+sr);
xs = aa*exp(ee./tk);

ak1 = a1*exp(-(e1./tk));
inters = zeros(size(ak1));
intere = zeros(size(ak1));
delt = zeros(size(ak1));
portions = zeros(size(ak1));

for i=1:length(inters)
    if ismember(i,pos)
        inters(i)=0;
        delt(i)=0;
        portions(i) = 0;
    else
        if intere(i-1)<1
            inters(i) = intere(i-1);
        else
            inters(i) = intere(i-1)*(1-xi(i));
        end

    end
    intere(i)= xs(i)-(xs(i)-inters(i))*exp(-ak1(i));

    if ~ismember(i,pos)
        if intere(i)<1
            delt(i) = 0;
        else
            delt(i) = xi(i)*intere(i);
```

```
        end
        portions(i) = delt(i) + portions(i-1);
    end

end

end
```

Code S2. Function *utah* MATLAB code, for calculating the chilling units according to the Utah model (Richardson et al., 1974)

```
function [ct] = utah(temperature)

ct = zeros(length(temperature),1);


%if 1.4<Temperature<=2.4
x = temperature>1.4 & temperature<=2.4;
ct(x) = ct(x)+0.5;

%if 2.4<Temperature<=9.1
x = temperature>2.4 & temperature<=9.1;
ct(x) = ct(x)+1;

%if 9.1<Temperature<=12.4
x = temperature>9.1 & temperature<=12.4;
ct(x) = ct(x)+0.5;

%if 12.4<Temperature<=15.9 multiplies by 0

%if 15.9<Temperature<=18
x = temperature>15.9 & temperature<=18;
ct(x) = ct(x)-0.5;

%if Temperature>18
x = temperature>18;
ct(x) = ct(x)-1;
```

Code S3. Function *north_caroline* MATLAB code, for calculating the chilling units according to the North Caroline model (Shaltout and Unrath, 1983)

```
function [ct] = noth_caroline(temperature)

% North Caroline model (from Shaltout and Unrath 1983, pg 959)

ct = zeros(length(temperature),1);
%if -1.1=<Temperature, zero contribution

%if -1.1<Temperature<=1.6
x = temperature>-1.1 & temperature<=1.6;
ct(x) = ct(x)+0.5;

%if 1.6<Temperature<=7.2
x = temperature>1.6 & temperature<=7.2;
ct(x) = ct(x)+1;

%if 7.2<Temperature<=13
x = temperature>7.2 & temperature<=13;
ct(x) = ct(x)+0.5;
```

```
%if 13<Temperature<=16.5, zero contribution

%if 16.5<Temperature<=19
x = temperature>16.5 & temperature<=19;
ct(x) = ct(x)-0.5;

%if 19<Temperature<=20.7
x = temperature>19 & temperature<=20.7;
ct(x) = ct(x)-1;

%if 20.7<Temperature<=22.1
x = temperature>20.7 & temperature<=22.1;
ct(x) = ct(x)-1.5;

%if 22.1>Temperature
x = temperature>22.1;
ct(x) = ct(x)-2;
```

Code S4. Function *melo_abreu* MATLAB code, for calculating the chilling units according to the de Melo-Abreu model (De Melo-Abreu et al., 2004)

```
function [ct] = melo_abreu(temperature)

% Model 1 for Olives (Melo-Abreu, fig. 1 and section 3.1.)

ct = zeros(length(temperature),1);

%Define constants
%To: optimum temperature for chilling (°C)
To=7.3;
%Tx: breakpoint temperature (°C)
Tx = 20.7;
%a: chilling units nullify when temperature is above Tx
a = -0.56;

%if 0<Temperature<=To
x = temperature>0 & temperature<=To;
ct(x) = ct(x)+temperature(x)/To;

%if To<Temperature<=Tx
x = temperature>To & temperature<=Tx;
ct(x) = ct(x) + 1 - (temperature(x)-To)*(1-a)/(Tx-To);

%if Temperature>Tx
x = temperature>Tx;
ct(x) = ct(x) + a;
```

Code S5. Function *hour_temp* MATLAB code for calculating hourly temperature from maximum and minimum daily temperatures following the de Wit et al. (1978)de Wit et al. (1978) approach.

```
function [thour,time] = hour_temp(Tmax,Tmin,dates,lat)

% Calculates the temperature for each hour of the day taking into account
% the maximum and minimum temperatures

% Input parameters
    % Tmax: vector with maximum temperatures for each day
```

```
    % Tmin: vector with minimum temperatures for each day
    % dates: vector with the date for each row
    % lat: latitude of the cell

% Output parameters
    % thour: temperature for each hour
    % time: vector with the date and time for each row

%constants
hour_col = 4;
day_col = 3;
month_col = 2;
year_col = 1;

%creates the array of days, with values from 1 to 366
d = zeros(length(dates),1);
for i =1 : length(dates)
    %it restarts the count with each new year
    if(dates(i,month_col)==1 && dates(i,day_col)==1)
        d(i) = 1;
    else
        d(i) = d(i-1)+1;
    end
end

%creates Tmax_aux and Tmin_aux, as auxiliar variables for calculations
Tmax_aux = zeros(length(Tmax)+1,1);
Tmax_aux(1) = Tmax(1);
Tmax_aux(2:end) = Tmax;
Tmin_aux = zeros(length(Tmin)+1,1);
Tmin_aux(end) = Tmin(end);
Tmin_aux(1:(end-1)) = Tmin;

% gets the sunrise hour (tr) for each day (d)
[tr] = sunrise_time(d,lat);

% creates the array for saving hour temperatures
thour = zeros(23,length(d));
hour = zeros(23,length(d));
years = zeros(23,length(d));
months = zeros(23,length(d));
days = zeros(23,length(d));

%gets the temperature of all days for each hour (0-23)
for t=0:23

    %if t<tr
    x = find (t<tr);
    if ~isempty(x)
        thour(t+1,x) = (Tmax_aux(x) + Tmin(x))/2 + ...
            (Tmax_aux(x)-Tmin(x))/2.*cos(pi*(t+10)./(tr(x)+10));
    end
    %if t>=tr && t<14
    x = find(t>=tr & t<14);
    if ~isempty(x)
    thour(t+1,x) = (Tmax(x) + Tmin(x))/2 - ...
        ((Tmax(x)-Tmin(x))/2).*cos(pi*(t-tr(x))./(14-tr(x)));
    end
    %if t>=14 && t<24
    if t>=14
        thour(t+1,:) = (Tmax + Tmin_aux(2:end))/2 + (Tmax-
Tmin_aux(2:end))/2.*cos(pi*(t-14)./(tr+10));
    end
```

```
    hour(t+1,:) = t*ones(length(d),1);
    years(t+1,:) = dates(:,year_col);
    months(t+1,:) = dates(:,month_col);
    days(t+1,:) = dates(:,day_col);
end

%transforms the thour array into a vector
thour = thour(:);
time = zeros(length(thour),6);
time(:,year_col) = years(:);
time(:,month_col) = months(:);
time(:,day_col) = days(:);
time(:, hour_col) = hour(:);
```

Code S6. Function *sunrise_time* MATLAB code for calculate the sunrise time for each day from latitude

```
function [h] = sunrise_time(d,Lat)

% calculates the sunrise time for each day (d) taking
% into account the latitude

% Input parameters
    % d: matrix with the days of year
% Output parameters
    % h: sunrise time for each day in the input matrix

%D: day of year in degrees
D = 360*d/365;
%dec: declination in degrees
dec = -23.5*cosd(D+9.865);
%w: angle at sunrise in degrees
w=acosd(-tand(Lat)*tand(dec));
%h: solar time in hour
h = 12-w/15;
```

Code S7. Function *calculate_period* MATLAB code for calculate initial and final chilling accumulation periods used for all chilling models but the Dynamic one.

```
% This function takes the chilling accumulation and checks the "area"
% that would remain taking the maximum values (relatives). It is like if we
% trace a horizontal line, see where it intersects and we fill the
% resulting area. The biggest area is the chosen one. Then, the minimum
% is identified as the one laying in that region.

function [ pos_minimo, pos_maximo ] = calculate_period(d)

acc = d;
dlen = length(d);

% Remove the first part where only go down
baja = 1;
c = 0;
while baja>0
   c = c + 1;

   % If arrives to the end means it never went down
   if c == dlen
      c = 1;
```

```
      baja = 0;
      break;
   end

   % Does not go down
   if d(c+1)>d(c)
       baja = 0;
   end
end
for i=1:(c-1)
  d(i) = NaN;
end


x = isnan(d);
numnan = length(d(x));
d = d(~x);


% ListaEnlazada is just a linked list data structure
CORTEs = ListaEnlazada();
MAXs = ListaEnlazada();
AREAs = ListaEnlazada();

anterior = d(1);
for i=2:(dlen-numnan)-1

      if is_maximum(d,i)>0

        POs = find_left_cut(acc,d(i),i+numnan);

        corte = 1;
        if length(POs)>0
            % Remove the same point
            for j=1:length(POs)
               if POs(j)==(i+numnan)
                   POs(j) = NaN;
               end
            end
            x = isnan(POs);
            POs = POs(~x);

            if length(POs)==0
                corte = 1;
            else
                posaux = POs;
                for aa=1:length(POs)
                   pini = POs(aa);
                   pfin = i+numnan;

                   aux = acc(pini:pfin);
                   aux = max(aux);

                   % It does not count because it cut the graph
                   if acc(pini)<aux
                      x = posaux == POs(aa);
                      posaux = posaux(~x);
                   end
                end
                POs = posaux;

                dist = POs-(i+numnan);
```

```
                x = dist == min(dist);
                corte = POs(x);
                if length(corte)>1
                    corte = corte(1);
                end
            end
        end

        p1 = corte;
        p2 = i+numnan;

        area = 0;
        for j=p1:p2
           area = area + abs(max(d(i),acc(j))-min(d(i),acc(j)));
        end

        % Add to the end of the linked list
        MAXs.insertaFinal(i+numnan);
        CORTEs.insertaFinal(p1);
        CORTEs.insertaFinal(p2);
        AREAs.insertaFinal(area);

        close all
    end

    anterior = d(i);
end


% Get number matrix from linked list
MAXs = MAXs.getMatrizNumerica();
CORTEs = CORTEs.getMatrizNumerica();
AREAs = AREAs.getMatrizNumerica();

if length(MAXs)>0

    x = AREAs == max(AREAs);
    p = find(x);
    if length(p)>0
        p=p(1);
    end

    % Search for the minimum
    posmin = CORTEs((p-1)*2+1);
    minim = acc(CORTEs((p-1)*2+1));
    for j=(CORTEs((p-1)*2+1)):(CORTEs((p-1)*2+2))
        if acc(j)<minim
            minim = acc(j);
            posmin = j;
        end
    end

    pos_minimo = posmin;
    pos_maximo = MAXs(p);

else
    pos_minimo = NaN;
    pos_maximo = NaN;
end

end
```

Code S8. Function *is_maximum* MATLAB code, auxiliary for *calculate_period* function

```
% Given an array with values, it tell us if the value in the position i is a
% relative maximum. It will be considered also maximum if it is the last one of
% a series of the same values

function r = is_maximum( d, i )

if i<=1
    r = 0;
    return;
end

ok = 0;
if i==length(d)
    ok = 1;
else
    if d(i)>d(i+1)
        ok = 1;
    end
end

% If its the last one or is bigger than the next one it can be maximum
if ok>0
   if d(i)>d(i-1)
       r = 1;
       return;
   end
   aux = i;
   while d(aux) == d(i)
       aux = aux - 1;
   end
   if d(i)>d(aux)
       r = 1;
       return;
   else
       r = 0;
       return;
   end

else
    r = 0;
    return;
end
end
```

Code S9. Function *find_left_cut* MATLAB code, auxiliary for *calculate_period* function

```
% It finds the positions x where the plot intersects with the horizontal line y = v
% (only cuts at the left of x) y = f(x)

function [ POs ] = find_left_cut(d,v,x)

POs = ListaEnlazada();
for i=1:length(d)-1
   if  (d(i) > v && d(i+1) < v) || ...
           (d(i) < v && d(i+1) > v)
        POs.insertaFinal(i);
        POs.insertaFinal(i+1);
   else
```

```
        if d(i) == v
          POs.insertaFinal(i);
        else if d(i+1) == v
          POs.insertaFinal(i+1);
            end
        end
   end
end

% Delete repeated
POs = POs.eliminaRepetidos();
aux = x-1;
while aux>0 && d(aux)==d(x)
    posaux = POs.posiciones(aux);
    posaux = posaux.getMatrizNumerica();
    posaux = posaux(1);

    % Delete element in position posaux
    POs.eliminaN(posaux);
    aux = aux - 1;
end

% Convert to numerical matrix and we left only the left ones
POs = POs.getMatrizNumerica();
xxx = POs<x;
POs = POs(xxx);
```

References

De Melo-Abreu, J. P., Barranco, D., Cordeiro, A. M., Tous, J., Rogado, B. M., and Villalobos, F. J.: Modelling olive flowering date using chilling for dormancy release and thermal time, Agric. For. Meteorol., 125, 117-127, 10.1016/j.agrformet.2004.02.009, 2004.

de Wit, C. T., Goudriaan, J., van Laar, H. H., Penning de Vries, F. W. T., Rabbinge, R., Van Keulen, H., Louwerse, W., Sibma, L., and de Jonge, C.: Simulation of assimilation, respiration and transpiration of crops, Pudoc, Wageningen, 1978.

Fishman, S., Erez, A., and Couvillon, G. A.: The temperature dependence of dormancy breaking in plants: Mathematical analysis of a two-step model involving a cooperative transition, Journal of Theoretical Biology, 124, 473-483, https://doi.org/10.1016/S0022-5193(87)80221-7, 1987a.

Fishman, S., Erez, A., and Couvillon, G. A.: The temperature dependence of dormancy breaking in plants: Computer simulation of processes studied under controlled temperatures, 309-321 pp., 1987b.

Giorgi, F., and Gutowski, W. J.: Regional Dynamical Downscaling and the CORDEX Initiative, Annual Review of Environment and Resources, 40, 467-490, 10.1146/annurev-environ-102014-021217, 2015.

Luedeling, E., and Brown, P.: A global analysis of the comparability of winter chill models for fruit and nut trees, 411-421 pp., 2011.

Richardson, E. A., Seeley, S. D., and Walker, D. R.: A model for estimating the completion of rest for 'Redhaven' and 'Elberta' peach trees, HortScience, 1, 331-332, 1974.

Shaltout, A. D., and Unrath, C. R.: Rest completion prediction model for 'Starkrimson Delicious' apples, J. Amer. Soc. Hort. Sci., 108, 957-961, 1983.

Taylor, K. E., Stouffer, R. J., and Meehl, G. A.: An Overview of CMIP5 and the Experiment Design, Bull. Amer. Meteorol. Soc., 93, 485-498, 10.1175/bams-d-11-00094.1, 2012.