Supplementary material: MATLAB code

This supplementary material provides the Matlab code used for calculating the results presented in the paper.

```
function
[Densities,Distance,SARCAMat,ProductSum,M,MAbs,Countries,CountryCompInd,Pro
ductCompInd,CP,Prox,Centrality,DistanceAndOpporGain,GVCFull,GVCAct,GVCTier,
P2AllP,P2P,A2A,T2T,PercInGVCProd,PercInGVCAct,PercInGVCTier,P2AllP_Vec,P2P_
Vec,BaseLineMetrics,BaseRCASpace,GVCFullBaseRCA,DistReq3,Comp3,OpporGainSce
narios3,BestFrom3Comp,BestFrom3Oppor,GVCRCAScenarios1,GVCRCAScenarios2,GVCR
CAScenarios3,ProxSums,Products,BestComplexityContr,SumCompContr,NumberInAct
ToBeAdded,tS1_Seconds,tS2_Seconds,tS3_Minutes] =
IO_PS_Code(All2014Mat,All2014CA,SARaw)

%This code executes the calculations as reported in the article:
%Bam, W., & De Bruyne, K. (2018). "Improving industrial policy
%intervention: the case of steel in South Africa", Journal of Development
%Studies. https://doi.org/10.1080/00220388.2018.1528354

%The code is written for Matlab.

%Version: 1.0
%Date: 22 August 2018
%Authors: Wouter Bam and Karolien De Bruyne

%If any errors are identified, please email wouterb@sun.ac.za and CC
%woutergb@gmail.com

%The code requires the loading into the Matlab Workspace the Raw World
%Trade Data in Matrix Form and Cell array Form (already in alphabetical
%order) and the Raw Country Data in Matrix Form.

%Both of these are contained in the saved workspace called:
%"AllRequiredInputs.mat" which can provided on request.

%The variables in this workspace (downloaded from the OEC website) are as
follows:

%All2014Mat; % Format = ( 1 Year; 2 Country; 3 hs92code; 4 Export Value; 5
Import value: 6 Export RCA; 7 Import RCA)
%All2014CA;  % Format = ( 1 Year; 2 Country; 3 hs92code; 4 Export Value; 5
Import value: 6 Export RCA; 7 Import RCA)
%SARaw;  % Format = ( 1 Year; 2 Country; 3 hs92code; 4 Export Value; 5
Import value: 6 Export RCA; 7 Import RCA)

%The code also requires a csv file ("VC Mapping.csv")
%with the GVC mapping in three columns in the format:
%1) Tier; 2) Product Category (referred to in the code as activity); and
%3) HS code of product.

%The code is ordered into various functions. This function
%(CaseStudyReworked) is the main function. It calls all the subfunctions
%required (provided below). The code outputs all relevant calculations as
%text files.
```

```
MainTime = tic; %Start timer

%% Declarations (Changeable inputs)
NumIters = 18; %The number of iterations used for the methods of reflection
used to calculate complexity

%% UpFront calculations:
Products = unique(All2014Mat(:,3)); %Drawing from the data all the product
codes to be used

%% Tranform country data to create country RCA matrix and sum world
production

%Outputs:
%SARCAMat   % Format = ( 1 hs92code; 2 Export of Country; 3 World Export of
product; 4 SA Good Percentage of world export of good; 5 % SA Good
Percentage of SA exports; 6  Good Percentage of world exports; 7 SA RCA)
%ProductSum % Format = 1 hs92 code; 2 World exports of product

[SARCAMat,ProductSum] = TransformCountry(All2014Mat,SARaw,Products);

%% Generate M matrixes

[M,MAbs,Countries] = generateM(Products,All2014Mat,All2014CA);

%% Calculate complexity

%CountryCompInd 1) complexities
%ProductCompInd 1) complexities

[CountryCompInd,ProductCompInd] =
CalcComplexity(M,Products,Countries,NumIters);

%% Calculate conditional probability and proximity matrix

%This code generates:
%1)CP(i,j) -> the conditional probability matrix.
%       It indicates the probability of an RCA in j given an RCA in i.
%       Its size is CP(Products,Products)
%2)Prox(i,j) -> the matrix of product proximities
%3)Centrality(i,2) a Matrix of ProductCentrality per product (1)
%HsCode; 2) Centrality )

[CP,Prox,Centrality] = GenerateCPandProxMat(M,Products,Countries);

%% Calculate Opportunity Gain and Distance

%RCA %Format = ( 1 hs92code; 2 SA RCA)

RCA = SARCAMat(:,[1 7]);

%DistanceAndOpporGain 1) HsCode; 2) Distance; 3) Distance if opportunity;
4) OpporGain; 5) OpporGain if Opportunity 6) Density; 7) Density if Oppor
[DistanceAndOpporGain,Densities,Distance,ProxSums] =
DistanceAndGain(Prox,RCA,Products,ProductCompInd);
```

```
%% Calculate GVC values

GVCMapping = csvread('VC Mapping.csv'); %GVCData 1) Tier#; 2) GVC
Activity#; 3) HS Code

[GVCFull,GVCAct,GVCTier] =
GVCDistanceAndGain(DistanceAndOpporGain,ProductCompInd,GVCMapping,Products,
SARCAMat,Densities);

dlmwrite('GVCTierResults.txt',GVCTier,'precision',10);
dlmwrite('GVCActResults.txt',GVCAct,'precision',10);
dlmwrite('GVCFullResults.txt',GVCFull,'precision',10);

[P2AllP,P2P,A2A,T2T,PercInGVCProd,PercInGVCAct,PercInGVCTier] =
CalcGVCProximities(GVCFull,GVCAct,GVCTier,Prox,Products);

dlmwrite('GVCProdToAllProductProx.txt',P2AllP,'precision',10);
dlmwrite('GVCProdToGVCProductProx.txt',P2AllP,'precision',10);
dlmwrite('GVCActToGVCActProx.txt',A2A,'precision',10);
dlmwrite('GVCTierToGVCTierProx.txt',T2T,'precision',10);

dlmwrite('AllProducts.txt',Products,'precision',10);
dlmwrite('GVCProducts.txt',GVCFull(:,3),'precision',10);

%% Calculate GVC Proximities

P2AllP_Vec = reshape(P2AllP,[],1);
P2P_Vec = reshape(P2P,[],1);

%% Set baseline for dynamic model

%BaseLineMetrics = zeros(9,1); % 1) Opportunity Value; 2) Average
complexity of products within the value chain with RCA > 1; 3) Avg distance
to Products in the VC with RCA < 1; 4) Average complexity of products
within the PS with RCA > 1; 5) Average distance to products in the product
space with RCA <1; 6) Num in GVC; 7) Sum of complexity in GVC; 8) Num in
PS; 9) Sum of complexity in PS

[BaseLineMetrics] =
CalcBaselineMetrics(Prox,RCA,Products,ProductCompInd,ProxSums,GVCFull,Dista
nceAndOpporGain,SARCAMat);

dlmwrite('BaseLineMetrics.txt',BaseLineMetrics,'precision',10);

%% Calculate RCA Space For Base

BaseRCASpace = CalculateBaseRCASpace(RCA); %Normalise current Product space

[GVCFullBaseRCA] = CalculateGVCFullBase(GVCFull); %Normalise current GVC
Full

NumProductsCurInGVC = BaseLineMetrics(6);
NumProductsCurInPS = BaseLineMetrics(8);

CurrentSumOfComplexityInGVC = BaseLineMetrics(7);
CurrentSumOfComplexityInPS = BaseLineMetrics(9);
```

```
%% Calculate Scenario RCA Space for 1

Scenario1 = tic;

[GVCRCAScenarioCont] =
CalcGVCRCASpaceScenarioCont(GVCFull,GVCAct,Products); % Calculate raw
contributions for each activity

[GVCFullRCAScenarioCont] = CalcGVCFullRCAScenarioCont(GVCFull,GVCAct);  %
Calculate raw contributions for each activity

[GVCRCAScenarios1] =
CalcGVCRCASpaceScenarios1(GVCRCAScenarioCont,BaseRCASpace,GVCAct,Products);
% Calculate Scenarios by adding baseline and Scenario contributions

[GVCFullRCAScenarios1] =
CalcGVCFullRCAScenarios1(GVCFullRCAScenarioCont,GVCFullBaseRCA,GVCAct,GVCFu
ll); % Calculate Scenarios by adding baseline and Scenario contributions


%% Calculate Metrics for RCA scenarios after 1

[Comp1 SumCompContr NumberInActToBeAdded] =
CalcCompContr(GVCFull,GVCAct,NumProductsCurInGVC,CurrentSumOfComplexityInGV
C); % Calculate Complexity values for Different Activities based on the
unexploited items

dlmwrite('SumCompContr.txt',SumCompContr,'precision',10);
dlmwrite('NumberInActToBeAdded.txt',NumberInActToBeAdded,'precision',10);

ComplexityCalcsFor1 = 'Complete'

[DistReq1] = CalcDist1(GVCFull,GVCAct); %Calculate the distance to
different activities during first iter

OpporValScenarios1 =
CalcOppValScenarios1(Products,Prox,ProductCompInd,GVCRCAScenarios1,GVCAct,B
aseRCASpace,ProxSums); %Calculate OpporGain for different activities during
first iter

OpporGainScenarios1 = OpporValScenarios1 - BaseLineMetrics(1);

Round1Metrics = [Comp1 OpporGainScenarios1 DistReq1];

dlmwrite('Round1Metrics.txt',Round1Metrics,'precision',10);

%% Calculate winners after 1

ScenariosDistances = ones(36,1);

for i = 1:36

   ScenariosDistances(i) = 2*i-1;

end
```

```
[BestFrom1Comp BestFrom1Oppor BestComplexityContr BestOpporGain] =
CalcBestFrom1(Round1Metrics,ScenariosDistances,BaseLineMetrics);

dlmwrite('BestFrom1Comp.txt',BestFrom1Comp,'precision',10);
dlmwrite('BestFrom1Oppor.txt',BestFrom1Oppor,'precision',10);

CalculationsFor1Metrics = 'Complete'

tS1_Seconds = toc(Scenario1)

%% Calculate Scenario RCA Space for 2

Scenario2 = tic;

[GVCRCAScenarios2] =
CalcGVCRCASpaceScenarios2(GVCRCAScenarioCont,BaseRCASpace,GVCAct,Products);
% Calculate Scenarios by adding baseline and Scenario contributions

[GVCFullRCAScenarios2] =
CalcGVCFullRCAScenarios2(GVCFullRCAScenarioCont,GVCFullBaseRCA,GVCAct,GVCFu
ll); % Calculate Scenarios by adding baseline and Scenario contributions

%% Calculate Metrics for RCA Scenarios after 2

[Comp2] =
CalcCompContr2(GVCAct,NumProductsCurInGVC,CurrentSumOfComplexityInGVC,SumCo
mpContr,NumberInActToBeAdded); % Calculate Complexity values for Different
Activities based on the unexploited items

dlmwrite('CompContr2.txt',Comp2,'precision',10);

[DistReq2] =
CalcDist2(GVCFull,GVCAct,GVCFullRCAScenarios1,GVCRCAScenarios1,ProxSums,Dis
tReq1,Products,Prox); %Calculate the distance to different activities
during first iter

dlmwrite('DistReq2.txt',DistReq2,'precision',10);
dlmwrite('DistReq1.txt',DistReq1,'precision',10);

OpporValScenarios2 =
CalcOppValScenarios2(Products,Prox,ProductCompInd,GVCRCAScenarios2,GVCAct,B
aseRCASpace,ProxSums); %Calculate OpporGain for different activities during
first iter

OpporGainScenarios2 = OpporValScenarios2 - BaseLineMetrics(1);

%% Calculate winners after 2

[BestFrom2Comp BestFrom2Oppor BestComplexityContr BestOpporGain] =
CalcBestFrom2(Comp2,DistReq2,OpporGainScenarios2,ScenariosDistances,BestFro
m1Comp,BestFrom1Oppor,BestComplexityContr,BestOpporGain);

dlmwrite('BestFrom2Comp.txt',BestFrom2Comp,'precision',10);
dlmwrite('BestFrom2Oppor.txt',BestFrom2Oppor,'precision',10);

CalculationsFor2Metrics = 'Complete'
```

```
tS2_Seconds = toc(Scenario2)

%% Calculate Scenario RCA Space for 3

Scenario3 = tic;

[GVCRCAScenarios3] =
CalcGVCRCASpaceScenarios3(GVCRCAScenarioCont,BaseRCASpace,GVCAct,Products);
% Calculate Scenarios by adding baseline and Scenario contributions

[GVCFullRCAScenarios3] =
CalcGVCFullRCAScenarios3(GVCFullRCAScenarioCont,GVCFullBaseRCA,GVCAct,GVCFu
ll); % Calculate Scenarios by adding baseline and Scenario contributions

%% Calculate Metrics for RCA Scenarios after 3

[Comp3] =
CalcCompContr3(GVCAct,NumProductsCurInGVC,CurrentSumOfComplexityInGVC,SumCo
mpContr,NumberInActToBeAdded); % Calculate Complexity values for Different
Activities based on the unexploited items

dlmwrite('CompContr3.txt',Comp3(3,3,:),'precision',10);

[DistReq3] =
CalcDist3(GVCFull,GVCAct,GVCFullRCAScenarios2,GVCRCAScenarios2,ProxSums,Dis
tReq1,DistReq2,Products,Prox); %Calculate the distance to different
activities during first iter

dlmwrite('DistReq3.txt',DistReq3(3,3,:),'precision',10);

OpporValScenarios3 =
CalcOppValScenarios3(Products,Prox,ProductCompInd,GVCRCAScenarios3,GVCAct,B
aseRCASpace,ProxSums); %Calculate OpporGain for different activities during
first iter

OpporGainScenarios3 = OpporValScenarios3 - BaseLineMetrics(1);

CalculationsFor3Metrics = 'Complete'

%% Calculate winners after 3

[BestFrom3Comp BestFrom3Oppor BestComplexityContr BestOpporGain] =
CalcBestFrom3(Comp3,DistReq3,OpporGainScenarios3,ScenariosDistances,BestFro
m2Comp,BestFrom2Oppor,BestComplexityContr,BestOpporGain);

dlmwrite('BestFrom3Comp.txt',BestFrom3Comp,'precision',10);
dlmwrite('BestFrom3Oppor.txt',BestFrom3Oppor,'precision',10);

%% Calculate metrics of winners

[MetricsForWinnersComp MetricsForWinnersOppor] =
CalcMetricsOfWinner(BestFrom3Comp,BestFrom3Oppor,GVCRCAScenarios3,GVCRCASce
narios2,GVCRCAScenarios1,Prox,ProxSums,Products,GVCFull);

dlmwrite('MetricsForWinnersComp.txt',MetricsForWinnersComp,'precision',10);
dlmwrite('MetricsForWinnersOppor.txt',MetricsForWinnersOppor,'precision',10
);
```

```
tS3_Minutes = toc(Scenario3) / 60
tMain = toc(MainTime) / 60

end

function [SARCAMat,ProductSum] =
TransformCountry(All2014Mat,CountryRaw,Products)

%This function transforms the raw data to calculate the country RCA
%It also Sums The world production for a specific good and calculates the
%RCA for the involved countries

% The resulting Arrays are as follows:
SARCAMat = zeros(size(Products,1),7); % Final format = ( 1 hs92code; 2
Export of Country; 3 World Export of product; 4 SA Good Percentage of world
export of good; 5 % SA Good Percentage of SA exports; 6  Good Percentage of
world exports; 7 SA RCA)
ProductSum = zeros(size(Products,1),2); %Final format = 1 hs92code; Sum of
world production

%All2014Mat; % Format = ( 1 Year; 2 Country; 3 hs92code; 4 Export Value; 5
Import value: 6 Export RCA; 7 Import RCA)
%CountryRaw; % Format =  ( 1 Year; 2 Country; 3 hs92code; 4 Export Value; 5
Import value: 6 Export RCA; 7 Import RCA)

Progress = 'TransformCountry_Start'

SARCAMat(:,1) = Products;
ProductSum(:,1) = Products;

for i = 1:size(Products,1) % run through all HS92s

    for j = 1:size(CountryRaw,1) %Run through all Country Exports

        if  CountryRaw(j,3) == SARCAMat(i,1)

            SARCAMat(i,2) = SARCAMat(i,2) + CountryRaw(j,4); %Add country
export to column 2

        end

    end

    for j = 1:size(All2014Mat,1) %Run through all raw world Exports entries

        if  All2014Mat(j,3) == ProductSum(i,1)

            ProductSum(i,2) = ProductSum(i,2) + All2014Mat(j,4); % Add up
world production to column 2 of Product Sum Matrix

        end

    end

end
```

```
SARCAMat(:,3) = ProductSum(:,2);

TotalSA = sum(SARCAMat(:,2));
TotalWorld = sum(ProductSum(:,2));

SARCAMat(:,4) = SARCAMat(:,2) ./ SARCAMat(:,3); % SA Good Percentage of
world export of good
SARCAMat(:,5) = SARCAMat(:,2) ./ TotalSA;  % SA Good Percentage of SA
exports
SARCAMat(:,6) = SARCAMat(:,3) ./ TotalWorld;  % Good Percentage of world
exports
SARCAMat(:,7) = SARCAMat(:,5) ./ SARCAMat(:,6); %SA RCA

Progress = 'TransformCountry_Finish'

end

function [M,MAbs,Countries] = generateM(Products,All2014Mat,All2014CA)

Progress = 'generateM_Start'

%This code generates the M and Mabs (with the actual RCA)

%IT requires the loading of:
%1) All2014CA; Format = % Format = ( 1 Year; 2 Country; 3 hs92code; 4
Export Value; 5 Import value: 6 Export RCA; 7 Import RCA) (It requires that
the countries are in alphabetical Order)
%2) All2014Mat; % Format = ( 1 Year; 2 Country; 3 hs92code; 4 Export Value;
5 Import value: 6 Export RCA; 7 Import RCA) (It requires that the countries
are in alphabetical Order)
%3) hs92codes;

CountriesDupl = All2014CA(:,2); %Reads all country names into array (Sorted
that names of countries are in alphabetical order)

Countries = unique(CountriesDupl); %Removes duplicates

ProductRCA = All2014Mat(:,[3 6]); %Filter only necessary parts of
All2014Mat (Sorted that names of countries are in alphabetical order)

MAbs = zeros(size(Countries,1),size(Products,1)); %Instantiate MAbs
M = zeros(size(Countries,1),size(Products,1)); %Instantiate M

c = 1; %country number

for k = 1:size(ProductRCA,1) %Run through all rows of All2014 array

    if k > 1

        if strcmp(CountriesDupl(k),CountriesDupl(k-1))

        else

            c=c+1;

        end
```

```
    end

    for j = 1:size(Products,1) %Run through the M array columns

        if Products(j) == ProductRCA(k,1)

            MAbs(c,j) = ProductRCA(k,2);

            if MAbs(c,j) > 1

                M(c,j) = 1;

            end

        end

    end

end

Progress = 'generateM_Finish'

end

function [CountryCompInd,ProductCompInd] =
CalcComplexity(M,Products,Countries,NumIters)

Progress = 'CalcComplexity_Start'

%This code generates the complexity of countries and products

%Output:
%CountryComplexityIndex6D(countries,complexities)
%ProductComplexityIndex6D(Products,complexities)
%Countries
%Products6D

%To Run the code it requires:
%1) M2D matrix (with 1's and 0's)
%2) Products2D
%3) Countries

M = M;

Kc0 = zeros(size(Countries,1),1); %Indicates the number of products each
country produces (array of diversity)

for i = 1:size(Countries,1)

    Kc0(i) = sum(M(i,:)) ;

end

Kp0 = zeros(size(Products,1),1); %Indicates the number of countries that
produce each product (array of ubiquity)

for i = 1:size(Products,1)
```

```
    Kp0(i) = sum(M(:,i)) ;

end


Kc1 = zeros(size(Countries,1),1); %Indicates the next iteration of
complexity of each country (array of country complexity 1)

for i = 1:size(Countries,1) %Run through countries

    ComplexitySum = 0;

    for j = 1:size(Products,1) %Run through products

        ComplexitySum = ComplexitySum + M(i,j)*Kp0(j);

    end

    Kc1(i) = ( 1/Kc0(i) ) * ComplexitySum;

end

Kp1 = zeros(size(Products,1),1); %Indicates the next iteration of
complexity products (array of product complexity 1)

for i = 1:size(Products,1) %Run through Products

    ComplexitySum = 0;

    for j = 1:size(Countries,1) %Run through Countries

        ComplexitySum = ComplexitySum + M(j,i)*Kc0(j);

    end

    Kp1(i) = ( 1 / Kp0(i) ) * ComplexitySum;

end

Kc2 = zeros(size(Countries,1),1); %Indicates the next iteration of
complexity of each country (array of country complexity 1)
Kp2 = zeros(size(Products,1),1); %Indicates the next iteration of
complexity products (array of product complexity 1)


for n = 2:NumIters  %Run through number of iterations as required

    KcLast = Kc1;
    KpLast = Kp1;

    for i = 1:size(Countries,1) %Run through countries

        ComplexitySum = 0;

        for j = 1:size(Products,1) %Run through products
```

```
            ComplexitySum = ComplexitySum + M(i,j)*Kp1(j);

        end

        Kc2(i) = ( 1 / Kc0(i) ) * ComplexitySum;


    end

    Kc1 = Kc2;

    for i = 1:size(Products,1) %Run through Products

        ComplexitySum = 0;

        for j = 1:size(Countries,1) %Run through Countries

            ComplexitySum = ComplexitySum + M(j,i)*Kc1(j);

        end

        Kp2(i) = ( 1 / Kp0(i) ) * ComplexitySum;

    end

    Kp1 = Kp2;

    n;

end

KcN = Kc2;
KpN = Kp2;

CountryCompInd = zeros(size(Countries,1),1);
ProductCompInd = zeros(size(Products,1),1);

for i = 1:size(Countries,1)

    CountryCompInd(i) = ( KcN(i) - mean(KcN(:)) ) / std(KcN(:));

end

for i = 1:size(Products,1)

    ProductCompInd(i) = ( KpN(i) - mean(KpN(:)) ) / std(KpN(:));

end

Progress = 'CalcComplexity_Finish'

end

function [CP,Prox,Centrality] = GenerateCPandProxMat(M,Products,Countries)
```

```
Progress = 'GenerateCPandProxMat_Start'

%This code generates:
%1)CP(i,j) -> the conditional probability matrix, .
%       It indicates the probability of an RCA in j given an RCA in i.
%       Its size is CP(Products,Products)

%2)Prox(i,j) -> the matrix of product proximities

%3)Centrality(i,2) a Matrix of ProductCentrality per product (1)
%HsCode; 2) Centrality )

%To Run the code it requires:
%1) M matrix (with 1's and 0's)
%2) Products
%3) Countries

CP = zeros(size(Products,1),size(Products,1)); %Instantiate CP

c = 1; %country number
p1 = 1; %product 1 (i) number
p2 = 1; %product 2 (j) number
CountP1 = 0;
CountP2GivenP1 = 0;

for p1 = 1:size(Products,1) %product 1 (i) number

    for p2 = 1:size(Products,1) %product 2 (j) number

        CountP1 = 0;
        CountP2GivenP1 = 0;

        for c = 1:size(Countries,1) %run through production of all
countries for product i and j

            if M(c,p1) == 1

                CountP1 = CountP1 + 1;

                if M(c,p2) == 1

                    CountP2GivenP1 = CountP2GivenP1 + 1;

                end

            end

        end

        CP(p1,p2) = CountP2GivenP1/CountP1;

    end

    p1;

end
```

```
Prox = CP; %Instantiate Prox equal to CP before minimum mirroring

for p1 = 1:size(Products,1) %product 1 (i) number

   for p2 = 1:size(Products,1) %product 2 (j) number

       if p1 ~= p2

            if Prox(p1,p2) > Prox(p2,p1)

                Prox(p1,p2) = Prox(p2,p1);

            else

                Prox(p2,p1) = Prox(p1,p2);

            end

       end

   end

   p1;

end

Centrality = zeros(size(Products,1),2); %Format = 1) Product; 2) Centrality

Centrality(:,1) = Products(:,1);

for i = 1:size(Products,1) %Run through all products

        Centrality(i,2) = (sum(Prox(i,:)) - 1)  / (size(Products,1) - 1) ;

end

Progress = 'GenerateCPandProxMat_Finish'

end


function [DistanceAndOpporGain,Densities,Distance,ProxSums] =
DistanceAndGain(Prox,RCA,Products,ProductCompInd)

%DistanceAndOpporGain:  %1) HsCode; 2) Distance; 3) Distance if
opportunity; 4) OpporGain; 5) OpporGain if Opportunity 6) Density; 7)
Density if Oppor

% Oppors  Format = (1 HSCodes; 2 Distances; 3 Distance if Unexploited; 4
RCA of Product; 5 Opportunity gain; 6 Adapted Opportunity gain; 7
Densities; 8 Densities if unexploited)

%It requires the loading of:
%1) Prox(Products,Products)
```

```
%2) SARCAMat ( 1 hs92code; 2 Export of Country; 3 World Export of prduct; 4
SA Good Percentage of world export of good; 5 % SA Good Percentage of SA
exports; 6  Good Percentage of world exports; 7 SA RCA)
%3) Products
%4) ProductCompInd

Products;
RCA;  % Format (1 hs92code; SA RCA of good)

Oppors = zeros(size(Products,1),8);
Oppors(:,1) = Products;
Oppors(:,4) = RCA(:,2);

Densities = zeros(size(Products,1),1);
Distance = zeros(size(Products,1),1);

CheckSumSA = zeros(size(Products,1),1);
OpenForestContrSum = zeros(size(Products,1),1);
ContributionToOpenForest = zeros(size(Products,1),1);
ContributionToOpenForestSAAdapted = zeros(size(Products,1),1);

%% Calculate Sums Of Proximities per product

ProxSums = zeros(size(Products,1),1);

for i=1:size(Products,1)

    ProxSums(i) = sum(Prox(i,:));

    i;

end

%% Calculate densities; Distance and Open Forest

%Calculate Total Opp value for country:

OpporValue = 0;

for i=1:size(Products,1) % Run through all potential products

    %Calculate Densities and Open Forest

    DensityNumerator = 0;
    DistanceNumerator = 0;

    for j=1:size(Products,1) %Run through all columns of the proximity
matrix

        if RCA(j,2) > 1 % If you are already competitive; calculate Density
from competitive to opportunity otherwise Distance and open forest

            if i ~= j % Make sure product is not not itself

                DensityNumerator = DensityNumerator + Prox(i,j);

            end
```

```
        else % Consider other products that could be unlocked from
opportunity

            if i ~= j % Make sure product is not not itself

                 DistanceNumerator = DistanceNumerator + Prox(i,j);

                if RCA(i,2) < 1 % Check if original activity is indeed an
opportunity

                    OpenForestContrSum(i) = OpenForestContrSum(i) +
Prox(i,j)*ProductCompInd(j)/(ProxSums(j) - 1);

                end

            end

        end

    end

    Densities(i) = DensityNumerator / (ProxSums(i) - 1);
    Distance(i) = DistanceNumerator / (ProxSums(i) - 1);

    ContributionToOpenForest(i) = OpenForestContrSum(i) - Densities(i)*
ProductCompInd(i);
    ContributionToOpenForestSAAdapted(i) = OpenForestContrSum(i);

    Oppors(i,2) = Distance(i,1);

    Oppors(i,5) = ContributionToOpenForest(i);
    Oppors(i,6) = ContributionToOpenForestSAAdapted(i);
    Oppors(i,7) = Densities(i);

    if RCA(i,2) <= 1 % If opportunity not exploited yet, add to unexploited
column

        Oppors(i,8) = Oppors(i,7);
        Oppors(i,3) = Oppors(i,2);

        OpporValue = OpporValue + Densities(i) * ProductCompInd(i);

    end

    i; %Keep track of execution

end

DistanceAndOpporGain = Oppors(:,[1,2,3,5,6,7,8]);  %1) HsCode; 2) Distance;
3) Distance if opportunity; 4) OpporGain; 5) OpporGain if Opportunity 6)
Density; 7) Density if Oppor

end

function [OpporValue] =
OpporValue(Prox,RCA,Products,ProductCompInd,ProxSums)
```

```
%DistanceAndOpporGain:  %1) HsCode; 2) Distance; 3) Distance if
opportunity; 4) OpporGain; 5) OpporGain if Opportunity 6) Density; 7)
Density if Oppor
% This code calculates the "OpenForest" / Opportunity value for a country

Densities = zeros(size(Products,1),1);

%% Calculate densities; Distance and Open Forest

%Calculate Total Opp value for country:

OpporValue = 0;

for i=1:size(Products,1) % Run through all potential products

    if RCA(i,2) <= 1 % If opportunity not exploited yet

        DensityNumerator = 0;

        for j=1:size(Products,1) %Run through all columns of the proximity
matrix

            if RCA(j,2) > 1 % If you are already competitive; calculate
Density from competitive to opportunity

                if i ~= j % Make sure product is not not itself

                    DensityNumerator = DensityNumerator + Prox(i,j);

                end

            end
        end

        Densities(i) = DensityNumerator / (ProxSums(i) - 1);


        %if RCA(i,2) <= 1 % If opportunity not exploited yet, add to
unexploited column

        OpporValue = OpporValue + Densities(i) * ProductCompInd(i);

    end

end


end

function [GVCFull,GVCAct,GVCTier] =
GVCDistanceAndGain(DistanceAndOpporGain,ProductCompInd,GVCData,Products,SAR
CAMat,Densities)

%This code sums the distance, OpporGain and Complexity index metrics for
each
```

```
%of the value chain steps. It also distinguishes parts of each VC activity
%for which a competitive advante already exists and for those tha are still
%opportunities.

%Output:
%GVCFull 1) Tier#; 2) GVC Activity#; 3) HS Code; 4 Complexity; 5) Density;
6) RCA (7) Complexity if opp; 8) Distance if Opportunity; 9) OpporGain if
Opportunity; 10)RCA if Opp; 11) SA Export of Activity; 12) World Export of
activity; 13) SA export if oppor; 14) World export if opp; 15) ProdPosition
%GVCAct 1) Tier#; 2) GVC Activity ;3) Avg Complexity; 4) Avg Density; 5)
Avg RCA; 6) Avg Complexity if Opp; 7) Avg Distance if Opportunity; 8) Avg
OpporGain if Opportunity; 9) Avg RCA if Opp; 10) SA export of Activity; 11)
World export of activity; 12) SA export if oppor; 13) World export if opp;
14) # of Products In Activity ; 15) # of Opp Products in Act
%GVCTier 1) Tier#; 2) Avg Complexity; 3) Avg Density; 4) Avg RCA; 5) Avg
Complexity if Opp; 6) Avg Distance if Opportunity; 7) Avg OpporGain if
Opportunity; 8) Avg RCA if Opp; 9) SA export of Activity; 10) World export
of activity; 11) SA export if oppor; 12) World export if opp; 13) Number
Act in Tier; 14) Number of Opp Act in Tier

%It requires
%1) DistanceAndOpporGain %1) HsCode; 2) Distance; 3) Distance if
opportunity; 4) OpporGain; 5) OpporGain if Opportunity 6) Density; 7)
Density if Oppor
%2) ProductCompInd
%3) GVCData 1) Tier#; 2) GVC Activity#; 3) HS Code
%4) Products
%5) SARCAMat 1) hs92code; 2) Export of Country; 3) World Export of product;
4) SA Good Percentage of world export of good; 5) SA Good Percentage of SA
exports; 6)  Good Percentage of world exports; 7) SA RCA
%6) Densities

NumberGVCTiers = size(unique(GVCData(:,1)),1);
NumberGVCActivities = size(unique(GVCData(:,2)),1);
NumberGVCProducts = size(unique(GVCData(:,3)),1);

NumberProductsInAct = zeros(NumberGVCActivities(1),1);
NumberActInTier = zeros(NumberGVCTiers(1),1);
NumberProdInTier = zeros(NumberGVCTiers(1),1);

NumberOppProductsInAct = zeros(NumberGVCActivities(1),1);
NumberOppActInTier = zeros(NumberGVCTiers(1),1);
NumberOppProdInTier = zeros(NumberGVCTiers(1),1);

GVCFull = zeros(NumberGVCProducts(1),15);
GVCAct = zeros(NumberGVCActivities(1),15);
GVCTier = zeros(NumberGVCTiers(1),14);

%% Concatenate

for i = 1:size(GVCData,1) %Step through all GVCData components

    HSCodePosition = find(Products==GVCData(i,3));

    GVCFull(i,15) = HSCodePosition; %Record link to product

    GVCFull(i,[1,2,3]) = GVCData(i,[1,2,3]); % Copy basic variables
    GVCFull(i,4) = ProductCompInd(HSCodePosition); %Copy complexity
    GVCFull(i,5) = Densities(HSCodePosition); %Copy Densities
```

```
    GVCFull(i,6) = SARCAMat(HSCodePosition,7); %Copy RCA

    GVCFull(i,[11,12]) = SARCAMat(HSCodePosition,[2,3]); %Copy SA and World
export of activity

    Act = GVCFull(i,2); %Check which activity is being dealt with
    NumberProductsInAct(Act,1) = NumberProductsInAct(Act,1) + 1; %Count
number of products in act

    if GVCFull(i,6) < 1 %Copy if opportunity.

        GVCFull(i,[7,10,13,14]) = GVCFull(i,[4,6,11,12]); %Copy complexity,
RCA if opportunity and export if opp
        GVCFull(i,[8,9]) = DistanceAndOpporGain(HSCodePosition,[3,5]);
%Copy Distance and opportunity Gain if opp
        NumberOppProductsInAct(Act) = NumberOppProductsInAct(Act) + 1;
%Count number of products in act

    end

end

GVCAct(:,14) = NumberProductsInAct(:,1);
GVCAct(:,15) = NumberOppProductsInAct(:,1);

%% Sum trade total and for if opportunity for weighting to sum to GVCACT
and GVCTier

for i = 1:size(GVCData,1) %Step through all GVCData components

    j = GVCFull(i,2); % Activity
    k = GVCFull(i,1); % Tier

    GVCAct(j,1) = k; %Populate Tier
    GVCAct(j,2) = j; %Populate activity
    GVCTier(k,1) = k;

    GVCAct(j,[10,11,12,13]) = GVCAct(j,[10,11,12,13]) +
GVCFull(i,[11,12,13,14]); %Copy trade
    GVCTier(k,[9,10,11,12]) = GVCTier(k,[9,10,11,12]) +
GVCFull(i,[11,12,13,14]); %Copy trade

end

%% Use trade weighting for summing to Act (Do not weight for trade)

for i = 1:size(GVCData,1) %Step through all GVCData components

    j = GVCFull(i,2); % Activity

    GVCAct(j,[3,4,5]) = GVCAct(j,[3,4,5]) + GVCFull(i,[4,5,6]) ./
NumberProductsInAct(j);  % Copy those that are average for all products
that are part of activity
    GVCAct(j,[6,7,8,9]) = GVCAct(j,[6,7,8,9]) + GVCFull(i,[7,8,9,10]) ./
NumberOppProductsInAct(j);  % Copy those that are average for opportunities
(weighted only according to opportunities' export)

end
```

```
%% Count number of products per tier

for i = 1:size(GVCFull,1)

    Tier = GVCFull(i,1);
    NumberProdInTier(Tier) = NumberProdInTier(Tier) + 1;

    RCAForProd = GVCFull(i,6);

    if RCAForProd < 1

        NumberOppProdInTier(Tier) = NumberOppProdInTier(Tier) + 1;

    end

end


%% Count number of activities per tier

for i = 1:size(GVCAct,1)

    Tier = GVCAct(i,1);
    NumberActInTier(Tier) = NumberActInTier(Tier) + 1;

    RCAForAct = GVCAct(i,5);

    if RCAForAct < 1

        NumberOppActInTier(Tier) = NumberOppActInTier(Tier) + 1;

    end

end

GVCTier(:,13) = NumberActInTier(:,1);
GVCTier(:,14) = NumberOppActInTier(:,1);

%% Use trade weighting for summing to Tier (Do not weight for trade)

for i = 1:size(GVCFull,1) %Step through all GVCFull components

    k = GVCFull(i,1); % Tier

    GVCTier(k,[2,3,4]) = GVCTier(k,[2,3,4]) + GVCFull(i,[4,5,6]) ./
NumberProdInTier(k);  % Copy those that are average for all products that
are part of activity
    GVCTier(k,[5,6,7,8]) = GVCTier(k,[5,6,7,8]) + GVCFull(i,[7,8,9,10])  ./
NumberOppProdInTier(k);  % Copy those that are average for opportunities
(weighted only according to opportunities' export)

end

end
```

```
function [P2AllP,P2P,A2A,T2T,PercInGVCProd,PercInGVCAct,PercInGVCTier] =
CalcGVCProximities(GVCFull,GVCAct,GVCTier,Prox,Products)

Progress = 'CalcGVCProximities_Start'

%This function evaluates the proximities between products that form part of
the GVC

%It generates various matrixes:
%P2AllP (Dimensions 1: GVCProd#; Dimensions 2: AllProducts#)(this array has
all the proximities of GVC products to all the other products)
%P2P (Dimensions 1: GVCProd#; Dimensions 2: GVCProd#)(this array has all
the proximities of GVC products to all the other GVC products )
%A2A (Dimensions 1: GVCAct#; Dimensions 2: GVCAct#)(this array has all the
proximities of GVC activities to all the other GVC activities)
%T2T (Dimensions 1: GVCTier#; Dimensions 2: GVCTier#)(this array has all
the proximities of GVC activities to all the other GVC activities)

%It requires the following matrixes:
%GVCFull 1) Tier#; 2) GVC Activity#; 3) HS Code; 4 Complexity; 5) Density;
6) RCA (7) Complexity if opp; 8) Distance if Opportunity; 9) OpporGain if
Opportunity; 10)RCA if Opp; 11) SA Export of Activity; 12) World Export of
activity; 13) SA export if oppor; 14) World export if opp; 15) ProdPosition
%GVCAct 1) Tier# 2) GVC Activity 3) Avg Complexity; 4) Avg Density; 5) Avg
RCA; 6) Avg Complexity if Opp; 7) Avg Distance if Opportunity; 8) Avg
OpporGain if Opportunity; 9) Avg RCA if Opp; 10) SA export of Activity; 11)
World export of activity; 12) SA export if oppor; 13) World export if opp;
14) # of Products In Activity ; 15) # of Opp Products in Act
%GVCTier 1) Tier# 2) Avg Complexity; 3) Avg Density; 4) Avg RCA; 5) Avg
Complexity if Opp; 6) Avg Distance if Opportunity; 7) Avg OpporGain if
Opportunity; 8) Avg RCA if Opp; 9) SA export of Activity; 10) World export
of activity; 11) SA export if oppor; 12) World export if opp; 13) Number
Act in Tier; 14) Number of Opp Act in Tier

%Prox(i,j) -> the matrix of product proximities
%Products

NumberGVCTiers = size(GVCTier,1);
NumberGVCActivities = size(GVCAct,1);
NumberGVCProducts = size(GVCFull,1);
NumberProducts = size(Products,1);

P2AllP = zeros(NumberGVCProducts,NumberProducts);
P2P = zeros(NumberGVCProducts,NumberGVCProducts);
A2A = zeros(NumberGVCActivities,NumberGVCActivities);
A2ANumer = zeros(NumberGVCActivities,NumberGVCActivities);
A2ADenom = zeros(NumberGVCActivities,NumberGVCActivities);
T2T = zeros(NumberGVCTiers,NumberGVCTiers);
T2TNumer = zeros(NumberGVCTiers,NumberGVCTiers);
T2TDenom = zeros(NumberGVCTiers,NumberGVCTiers);
ProductCodePositions = zeros(NumberGVCProducts,1);
AllProxAct = zeros(NumberGVCActivities,1);
AllProxTier = zeros(NumberGVCTiers,1);
ProxActInGVC = zeros(NumberGVCActivities,1);
ProxTierInGVC = zeros(NumberGVCTiers,1);
SumProxProductAll = zeros(NumberGVCProducts,1);
SumProxProductInGVC = zeros(NumberGVCProducts,1);
```

```
for i = 1:NumberGVCProducts

    ProductCodePositions(i) = find(Products==GVCFull(i,3));

end

%% Populate P2AllP, P2P and count Proximities within each product,
activitiy and tier in relation to total

for i = 1:NumberGVCProducts

    ActNum1 = GVCFull(i,2);
    TierNum1 = GVCFull(i,1);

    ProductCode1Pos =  ProductCodePositions(i); %Capture product code for
activity in GVC

    P2AllP(i,:) = Prox(ProductCode1Pos,:); %Transfer all proximities for
all products

    for j = 1:NumberGVCProducts %Transfer only proximities for products in
the GVC

       ProductCode2Pos = ProductCodePositions(j);

       P2P(i,j) = Prox(ProductCode1Pos,ProductCode2Pos);

    end

    SumProxProductAll(i) = sum(P2AllP(i,:)) - 1;
    SumProxProductInGVC(i) = sum(P2P(i,:)) - 1;

    AllProxAct(ActNum1) = AllProxAct(ActNum1) + SumProxProductAll(i);
    AllProxTier(TierNum1) = AllProxTier(TierNum1) + SumProxProductAll(i);

    ProxActInGVC(ActNum1) = ProxActInGVC(ActNum1) + SumProxProductInGVC(i);
    ProxTierInGVC(TierNum1) = ProxTierInGVC(TierNum1) +
SumProxProductInGVC(i);

end

%% Populate PercInGVC

PercInGVCProd = SumProxProductInGVC ./ SumProxProductAll; %Calculate the %
of proximities that are captured within the GVC
PercInGVCAct =  ProxActInGVC ./ AllProxAct; %Calculate the % of proximities
that are captured within the GVC
PercInGVCTier = ProxTierInGVC ./ AllProxTier; %Calculate the % of
proximities that are captured within the GVC

%% Populate A2A (Not weighting by trade)

for i = 1:NumberGVCProducts

    ActNum1 = GVCFull(i,2);
```

```
    for j = 1:NumberGVCProducts

        ActNum2 = GVCFull(j,2);

        A2ANumer(ActNum1,ActNum2) = A2ANumer(ActNum1,ActNum2) + P2P(i,j) ;
        A2ADenom(ActNum1,ActNum2) = A2ADenom(ActNum1,ActNum2) + 1 ;

    end

end

A2A = A2ANumer ./ A2ADenom;

%% Populate T2T

for i = 1:NumberGVCActivities

    TierNum1 = GVCAct(i,1);

    for j = 1:NumberGVCActivities

        TierNum2 = GVCAct(j,1);

        T2TNumer(TierNum1,TierNum2) = T2TNumer(TierNum1,TierNum2) +
A2A(i,j);
        T2TDenom(TierNum1,TierNum2) = T2TDenom(TierNum1,TierNum2) + 1;

    end

end

T2T = T2TNumer ./ T2TDenom;

Progress = 'CalcGVCProximities_Finish'


end

function [BaseLineMetrics] =
CalcBaselineMetrics(Prox,RCA,Products,ProductCompInd,ProxSums,GVCFull,Dista
nceAndOpporGain,SARCAMat)

BaseLineMetrics = zeros(9,1); % 1) Opportunity Value; 2) Average complexity
of products within the value chain with RCA > 1; 3) Avg distance to
Products in the VC with RCA < 1; 4) Average complexity of products within
the PS with RCA > 1; 5) Average distance to products in the product space
with RCA <1; 6) Num in GVC; 7) Sum of complexity in GVC; 8) Num in PS; 9)
Sum of complexity in PS

BaseLineMetrics(1) = OpporValue(Prox,RCA,Products,ProductCompInd,ProxSums);
[BaseLineMetrics(2),BaseLineMetrics(3),BaseLineMetrics(6),BaseLineMetrics(7
)] = VCCompAndDist(GVCFull);
[BaseLineMetrics(4),BaseLineMetrics(5),BaseLineMetrics(8),BaseLineMetrics(9
)] = PSCompAndDist(DistanceAndOpporGain,ProductCompInd,SARCAMat);

end
```

```
function [CompInVC,AvgDistToProdInVC,NumberInVC,SumCompInVC] =
VCCompAndDist(GVCFull)

%GVCFull 1) Tier#; 2) GVC Activity#; 3) HS Code; 4 Complexity; 5) Density;
6) RCA (7) Complexity if opp; 8) Distance if Opportunity; 9) OpporGain if
Opportunity; 10)RCA if Opp; 11) SA Export of Activity; 12) World Export of
activity; 13) SA export if oppor; 14) World export if opp; 15) ProdPosition

SumCompInVC = 0;
NumberInVC = 0;

SumDistToVC = 0;
NumberCurOutVC = 0;

for i = 1:size(GVCFull,1)

    if GVCFull(i,6) > 1 %Calculate avg complexity of products withing the
VC with RCA > 1

        SumCompInVC = SumCompInVC + GVCFull(i,4);
        NumberInVC = NumberInVC + 1;

    else % Calculate avg Distance to products in the VC with RCA < 1

        SumDistToVC = SumDistToVC + GVCFull(i,8);
        NumberCurOutVC = NumberCurOutVC + 1;

    end

end

CompInVC = SumCompInVC / NumberInVC;
AvgDistToProdInVC = SumDistToVC / NumberCurOutVC;

end

function [CompInPS,DistToPS,NumberWithRCA,SumCompInPS] =
PSCompAndDist(DistanceAndOpporGain,ProductCompInd,SARCAMat)

%1) DistanceAndOpporGain %1) HsCode; 2) Distance; 3) Distance if
opportunity; 4) OpporGain; 5) OpporGain if Opportunity 6) Density; 7)
Density if Oppor
%2) ProductCompInd
%3) SARCAMat 1) hs92code; 2) Export of Country; 3) World Export of product;
4) SA Good Percentage of world export of good; 5) SA Good Percentage of SA
exports; 6)  Good Percentage of world exports; 7) SA RCA

SumCompInPS = 0;
NumberWithRCA = 0;

SumDistToPS = 0;
NumberCurOutPS = 0;

for i = 1:size(ProductCompInd,1)

    if SARCAMat(i,7) > 1 %Calculate avg complexity of products within the
PS with RCA > 1
```

```
        SumCompInPS = SumCompInPS + ProductCompInd(i);
        NumberWithRCA = NumberWithRCA + 1;

    else % Calculate avg Distance to products in the PS with RCA < 1

        SumDistToPS = SumDistToPS + DistanceAndOpporGain(i,2);
        NumberCurOutPS = NumberCurOutPS + 1;

    end

end

CompInPS = SumCompInPS / NumberWithRCA;
DistToPS = SumDistToPS / NumberCurOutPS;

end

function [BaseRCASpace] = CalculateBaseRCASpace(RCA)

BaseRCASpace = zeros(size(RCA,1),1);
SetRCATo = 1.1;

for i = 1:size(RCA,1)

    if RCA(i,2) > 1

       BaseRCASpace(i,1) = SetRCATo;

    end

end

end

function [GVCFullBaseRCA] = CalculateGVCFullBase(GVCFull) %Normalise
current GVC Full

%GVCFull 1) Tier#; 2) GVC Activity#; 3) HS Code; 4 Complexity; 5) Density;
6) RCA (7) Complexity if opp; 8) Distance if Opportunity; 9) OpporGain if
Opportunity; 10)RCA if Opp; 11) SA Export of Activity; 12) World Export of
activity; 13) SA export if oppor; 14) World export if opp; 15) ProdPosition

GVCFullBaseRCA = zeros(size(GVCFull,1),1);
SetRCATo = 1.1;

for i = 1:size(GVCFull,1)

    if GVCFull(i,6) > 1

       GVCFullBaseRCA(i,1) = SetRCATo;

    end

end
```

```
end

function [GVCRCAScenarioCont] =
CalcGVCRCASpaceScenarioCont(GVCFull,GVCAct,Products)

%GVCFull 1) Tier#; 2) GVC Activity#; 3) HS Code; 4 Complexity; 5) Density;
6) RCA (7) Complexity if opp; 8) Distance if Opportunity; 9) OpporGain if
Opportunity; 10)RCA if Opp; 11) SA Export of Activity; 12) World Export of
activity; 13) SA export if oppor; 14) World export if opp; 15) ProdPosition

NumGVCAct = size(GVCAct,1);
NumProduct = size(Products,1);

GVCRCAScenarioCont = zeros(NumGVCAct,NumProduct);

SetRCATo = 1.1;

for i = 1:size(GVCFull,1)  %Run through all RCA's in GVCFull

    if GVCFull(i,6) < 1 %Consider All those for which RCA < 1 (i.e.
Populate all those in activity for which RCA not currently > 1)

        GVCRCAScenarioCont(GVCFull(i,2),GVCFull(i,15)) = SetRCATo;


    end

end

end

function [GVCFullRCAScenarioCont] =
CalcGVCFullRCAScenarioCont(GVCFull,GVCAct) % Calculate raw contributions
for each activity

%GVCFull 1) Tier#; 2) GVC Activity#; 3) HS Code; 4 Complexity; 5) Density;
6) RCA (7) Complexity if opp; 8) Distance if Opportunity; 9) OpporGain if
Opportunity; 10)RCA if Opp; 11) SA Export of Activity; 12) World Export of
activity; 13) SA export if oppor; 14) World export if opp; 15) ProdPosition

NumGVCAct = size(GVCAct,1);
NumInGVCFull = size(GVCFull,1);

GVCFullRCAScenarioCont = zeros(NumGVCAct,NumInGVCFull);

SetRCATo = 1.1;

for i = 1:size(GVCFull,1)  %Run through all RCA's in GVCFull

    if GVCFull(i,6) < 1 %Consider All those for which RCA < 1 (i.e.
Populate all those in activity for which RCA not currently > 1)

        GVCFullRCAScenarioCont(GVCFull(i,2),i) = SetRCATo;

    end

end
```

```
end

function [GVCRCAScenarios1] =
CalcGVCRCASpaceScenarios1(GVCRCAScenarioCont,BaseRCASpace,GVCAct,Products)

%GVCRCAScenarioCont: (NumGVCAct,NumProduct);
%BaseRCASpace = 1) RCA per product;

NumGVCAct = size(GVCAct,1);
NumProducts = size(Products,1);

GVCRCAScenarios1 = zeros(NumGVCAct,NumProducts);

for i = 1:NumGVCAct

    GVCRCAScenarios1(i,:) = transpose(BaseRCASpace) +
GVCRCAScenarioCont(i,:);

end

end

function [GVCFullRCAScenarios1] =
CalcGVCFullRCAScenarios1(GVCFullRCAScenarioCont,GVCFullBaseRCA,GVCAct,GVCFu
ll)

%GVCRCAScenarioCont: (NumGVCAct,NumProduct = RCA);
%BaseRCASpace = 1) RCA per product;

NumGVCAct = size(GVCAct,1);
NumInGVCFull = size(GVCFull,1);

GVCFullRCAScenarios1 = zeros(NumGVCAct,NumInGVCFull);

for i = 1:NumGVCAct

    GVCFullRCAScenarios1(i,:) = transpose(GVCFullBaseRCA) +
GVCFullRCAScenarioCont(i,:);

end

end

function [GVCRCAScenarios2] =
CalcGVCRCASpaceScenarios2(GVCRCAScenarioCont,BaseRCASpace,GVCAct,Products)

%GVCRCAScenarioCont: (NumGVCAct,NumProduct);
%BaseRCASpace = 1) RCA per product;

NumGVCAct = size(GVCAct,1);
NumProducts = size(Products,1);

GVCRCAScenarios2 = zeros(NumGVCAct,NumGVCAct,NumProducts);

for i = 1:NumGVCAct
```

```
    for j = 1:NumGVCAct

        GVCRCAScenarios2(i,j,:) = transpose(BaseRCASpace) +
GVCRCAScenarioCont(i,:) + GVCRCAScenarioCont(j,:);

    end

end

end

function [GVCFullRCAScenarios2] =
CalcGVCFullRCAScenarios2(GVCFullRCAScenarioCont,GVCFullBaseRCA,GVCAct,GVCFu
ll)

%GVCRCAScenarioCont: (NumGVCAct,NumProduct);
%BaseRCASpace = 1) RCA per product;

NumGVCAct = size(GVCAct,1);
NumInGVCFull = size(GVCFull,1);

GVCFullRCAScenarios2 = zeros(NumGVCAct,NumGVCAct,NumInGVCFull);

for i = 1:NumGVCAct

    for j = 1: NumGVCAct

        GVCFullRCAScenarios2(i,j,:) = transpose(GVCFullBaseRCA) +
GVCFullRCAScenarioCont(i,:) + GVCFullRCAScenarioCont(j,:);

    end

end

end

function [GVCRCAScenarios3] =
CalcGVCRCASpaceScenarios3(GVCRCAScenarioCont,BaseRCASpace,GVCAct,Products)

%GVCRCAScenarioCont: (NumGVCAct,NumProduct);
%BaseRCASpace = 1) RCA per product;

NumGVCAct = size(GVCAct,1);
NumProducts = size(Products,1);

GVCRCAScenarios3 = zeros(NumGVCAct,NumGVCAct,NumGVCAct,NumProducts);

for i = 1:NumGVCAct

    for j = 1:NumGVCAct

        for k = 1:NumGVCAct

            GVCRCAScenarios3(i,j,k,:) = transpose(BaseRCASpace) +
GVCRCAScenarioCont(i,:) + GVCRCAScenarioCont(j,:) +
GVCRCAScenarioCont(k,:);
```

```
        end

    end

end

end

function [GVCFullRCAScenarios3] =
CalcGVCFullRCAScenarios3(GVCFullRCAScenarioCont,GVCFullBaseRCA,GVCAct,GVCFu
ll)

%GVCRCAScenarioCont: (NumGVCAct,NumProduct);
%BaseRCASpace = 1) RCA per product;

NumGVCAct = size(GVCAct,1);
NumInGVCFull = size(GVCFull,1);

GVCFullRCAScenarios3 = zeros(NumGVCAct,NumGVCAct,NumGVCAct,NumInGVCFull);

for i = 1:NumGVCAct

    for j = 1:NumGVCAct

        for k = 1:NumGVCAct

            GVCFullRCAScenarios3(i,j,k,:) = transpose(GVCFullBaseRCA) +
GVCFullRCAScenarioCont(i,:) + GVCFullRCAScenarioCont(j,:) +
GVCFullRCAScenarioCont(k,:);

        end

    end

end

end

function [ComplexityInVCScenarios SumCompContr NumberInActToBeAdded] =
CalcCompContr(GVCFull,GVCAct,NumProductsCurInGVC,CurrentSumOfComplexityInGV
C)

%GVCFull 1) Tier#; 2) GVC Activity#; 3) HS Code; 4 Complexity; 5) Density;
6) RCA (7) Complexity if opp; 8) Distance if Opportunity; 9) OpporGain if
Opportunity; 10)RCA if Opp; 11) SA Export of Activity; 12) World Export of
activity; 13) SA export if oppor; 14) World export if opp; 15) ProdPosition

NumGVCAct = size(GVCAct,1);

SumCompContr = zeros(NumGVCAct,1); % 1) ContToComp (per activity)

NumberInActToBeAdded = zeros(NumGVCAct,1); % 1) NumInActivity (per
activity)

%ComplexityInVCScenarios = zeros(NumGVCAct,1); % 1) Complexity In VC for
scenario (per activity)
```

```
for i = 1:size(GVCFull,1)  %Run through all RCA's in GVCFull

    if GVCFull(i,6) < 1 %Consider All those for which RCA < 1 (i.e.
Populate all those in activity for which RCA not currently > 1)

        SumCompContr(GVCFull(i,2),1) = SumCompContr(GVCFull(i,2),1) +
GVCFull(i,4); %Sum complexity for activity
        NumberInActToBeAdded(GVCFull(i,2),1) =
NumberInActToBeAdded(GVCFull(i,2),1) + 1;

    end

end

ComplexityInVCScenarios = (SumCompContr + CurrentSumOfComplexityInGVC) ./
(NumberInActToBeAdded + NumProductsCurInGVC);

end

function [Comp2] =
CalcCompContr2(GVCAct,NumProductsCurInGVC,CurrentSumOfComplexityInGVC,SumCo
mpContr,NumberInActToBeAdded)

%GVCFull 1) Tier#; 2) GVC Activity#; 3) HS Code; 4 Complexity; 5) Density;
6) RCA (7) Complexity if opp; 8) Distance if Opportunity; 9) OpporGain if
Opportunity; 10)RCA if Opp; 11) SA Export of Activity; 12) World Export of
activity; 13) SA export if oppor; 14) World export if opp; 15) ProdPosition

NumGVCAct = size(GVCAct,1);

TotalSumCompCont = zeros(NumGVCAct,NumGVCAct,1); % 1) ContToComp (per
activity)

TotalNumberInActToBeAdded = zeros(NumGVCAct,NumGVCAct,1); % 1)
NumInActivity (per activity)

Comp2 = zeros(NumGVCAct,NumGVCAct,1); % 1) Complexity In VC for scenario
(per activity)

for i = 1:NumGVCAct

    for j = 1:NumGVCAct

        if i ~= j

            TotalSumCompCont(i,j) = SumCompContr(i) + SumCompContr(j);
            TotalNumberInActToBeAdded(i,j) = NumberInActToBeAdded(i) +
NumberInActToBeAdded(j);

            Comp2(i,j) = (TotalSumCompCont(i,j) +
CurrentSumOfComplexityInGVC) ./ (TotalNumberInActToBeAdded(i,j) +
NumProductsCurInGVC);

        end

    end
```

```
end

end

function [Comp3] =
CalcCompContr3(GVCAct,NumProductsCurInGVC,CurrentSumOfComplexityInGVC,SumCo
mpContr,NumberInActToBeAdded)

%GVCFull 1) Tier#; 2) GVC Activity#; 3) HS Code; 4 Complexity; 5) Density;
6) RCA (7) Complexity if opp; 8) Distance if Opportunity; 9) OpporGain if
Opportunity; 10)RCA if Opp; 11) SA Export of Activity; 12) World Export of
activity; 13) SA export if oppor; 14) World export if opp; 15) ProdPosition

NumGVCAct = size(GVCAct,1);

TotalSumCompCont = zeros(NumGVCAct,NumGVCAct,NumGVCAct,1); % 1) ContToComp
(per activity)

TotalNumberInActToBeAdded = zeros(NumGVCAct,NumGVCAct,NumGVCAct,1); % 1)
NumInActivity (per activity)

Comp3 = zeros(NumGVCAct,NumGVCAct,NumGVCAct,1); % 1) Complexity In VC for
scenario (per activity)

for i = 1:NumGVCAct

    for j = 1:NumGVCAct

        if i ~= j

            for k = 1:NumGVCAct

                if (k ~= j) &&  (k ~= i)

                    TotalSumCompCont(i,j,k) = SumCompContr(i) +
SumCompContr(j) + SumCompContr(k);
                    TotalNumberInActToBeAdded(i,j,k) =
NumberInActToBeAdded(i) + NumberInActToBeAdded(j) +
NumberInActToBeAdded(k);

                    Comp3(i,j,k) = (TotalSumCompCont(i,j,k) +
CurrentSumOfComplexityInGVC) ./ (TotalNumberInActToBeAdded(i,j,k) +
NumProductsCurInGVC);

                end

            end

        end

    end

end

end

function [DistReq1] = CalcDist1(GVCFull,GVCAct)
```

```
%GVCFull 1) Tier#; 2) GVC Activity#; 3) HS Code; 4 Complexity; 5) Density;
6) RCA (7) Complexity if opp; 8) Distance if Opportunity; 9) OpporGain if
Opportunity; 10)RCA if Opp; 11) SA Export of Activity; 12) World Export of
activity; 13) SA export if oppor; 14) World export if opp; 15) ProdPosition

NumGVCAct = size(GVCAct,1);

DistReq1 = zeros(NumGVCAct,1); % 1) DistReq1 (per activity)


for i = 1:size(GVCFull,1)  %Run through all RCA's in GVCFull

    if GVCFull(i,6) < 1 %Consider All those for which RCA < 1 (i.e.
Populate all those in activity for which RCA not currently > 1)

        DistReq1(GVCFull(i,2),1) = DistReq1(GVCFull(i,2),1) + GVCFull(i,8);
%Sum distances (only valid for first iter)

    end

end

end

function [DistReq2] =
CalcDist2(GVCFull,GVCAct,GVCFullRCAScenarios1,GVCRCAScenarios1,ProxSums,Dis
tReq1,Products,Prox)

%GVCFull 1) Tier#; 2) GVC Activity#; 3) HS Code; 4 Complexity; 5) Density;
6) RCA (7) Complexity if opp; 8) Distance if Opportunity; 9) OpporGain if
Opportunity; 10)RCA if Opp; 11) SA Export of Activity; 12) World Export of
activity; 13) SA export if oppor; 14) World export if opp; 15) ProdPosition

%GVCFullRCAScenarios1 = zeros(NumGVCAct,NumInGVCFull = RCA);
%GVCRCAScenarios1 = zeros(NumGVCAct,NumProducts);

NumGVCAct = size(GVCAct,1);
SizeGVCFull = size(GVCFull,1);

TotalDistances2 = zeros(NumGVCAct,NumGVCAct,1);
Distances2GVCFull = zeros(NumGVCAct,SizeGVCFull,1);

for ActAlreadyTaken = 1:NumGVCAct % Run through different scenarios from
which is being worked

    for i=1:size(GVCFull,1) % Run through all potential products in VC to
which distances must be calculated

        if GVCFullRCAScenarios1(ActAlreadyTaken,i) < 1  % If activity still
an opportunity, calculate distance to it.

            DistanceNumerator = 0;

            for j=1:size(Products,1) %Run through all columns of the
proximity matrix (i.e. all activities to the product in the VC)
```

```
                if GVCRCAScenarios1(ActAlreadyTaken,j) < 1 % If you do not
yet have a competitive advantage in other products, the distance increases

                    if GVCFull(i,15) ~= j % Make sure product is not not
itself

                        DistanceNumerator = DistanceNumerator +
Prox(GVCFull(i,15),j);

                    end

                end

            end

            Distances2GVCFull(ActAlreadyTaken,i) = DistanceNumerator /
(ProxSums(GVCFull(i,15)) - 1);

        end

    end

end

Distance2InVC = zeros(NumGVCAct,NumGVCAct,1);

for i = 1:NumGVCAct % Run through all scenarios

    for j = 1:SizeGVCFull % Run Through All Distances

        Distance2InVC(i,GVCFull(j,2)) = Distance2InVC(i,GVCFull(j,2)) +
Distances2GVCFull(i,j);

    end

end

for i = 1:NumGVCAct %Run through all scenario 1s

    for j = 1:NumGVCAct %Run through all scenario 2s

        TotalDistances2(i,j) = Distance2InVC(i,j) + DistReq1(i);

    end

end

DistReq2 = TotalDistances2;

end

function [DistReq3] =
CalcDist3(GVCFull,GVCAct,GVCFullRCAScenarios2,GVCRCAScenarios2,ProxSums,Dis
tReq1,DistReq2,Products,Prox)
```

```
%GVCFull 1) Tier#; 2) GVC Activity#; 3) HS Code; 4 Complexity; 5) Density;
6) RCA (7) Complexity if opp; 8) Distance if Opportunity; 9) OpporGain if
Opportunity; 10)RCA if Opp; 11) SA Export of Activity; 12) World Export of
activity; 13) SA export if oppor; 14) World export if opp; 15) ProdPosition

%GVCFullRCAScenarios1 = zeros(NumGVCAct,NumInGVCFull = RCA);
%GVCRCAScenarios1 = zeros(NumGVCAct,NumProducts);

NumGVCAct = size(GVCAct,1);
SizeGVCFull = size(GVCFull,1);

TotalDistances3 = zeros(NumGVCAct,NumGVCAct,NumGVCAct,1);
Distances3GVCFull = zeros(NumGVCAct,NumGVCAct,SizeGVCFull,1);

for Act1AlreadyTaken = 1:NumGVCAct % Run through different scenarios from
which is being worked

    for Act2AlreadyTaken = 1:NumGVCAct % Run through different scenarios
from which is being worked

        for i=1:size(GVCFull,1) % Run through all potential products in VC
to which distances must be calculated

            if GVCFullRCAScenarios2(Act1AlreadyTaken,Act2AlreadyTaken,i) <
1  % If activity still an opportunity, calculate distance to it.

                DistanceNumerator = 0;

                for j=1:size(Products,1) %Run through all columns of the
proximity matrix (i.e. all activities to the product in the VC)

                    if
GVCRCAScenarios2(Act1AlreadyTaken,Act2AlreadyTaken,j) < 1 % If you do not
yet have a competitive advantage in other products, the distance increases

                        if GVCFull(i,15) ~= j % Make sure product is not
not itself

                            DistanceNumerator = DistanceNumerator +
Prox(GVCFull(i,15),j);

                        end

                    end
                end

                Distances3GVCFull(Act1AlreadyTaken,Act2AlreadyTaken,i) =
DistanceNumerator / (ProxSums(GVCFull(i,15)) - 1);

            end

        end

    end

end
```

```
Distance3InVC = zeros(NumGVCAct,NumGVCAct,NumGVCAct,1);

for i = 1:NumGVCAct % Run through all scenarios

    for j = 1:NumGVCAct % Run through all scenarios

        for k = 1:SizeGVCFull % Run Through All Distances

            Distance3InVC(i,j,GVCFull(k,2)) =
Distance3InVC(i,j,GVCFull(k,2)) + Distances3GVCFull(i,j,k);

        end

    end

end


for i = 1:NumGVCAct %Run through all scenario 1s

    for j = 1:NumGVCAct %Run through all scenario 2s

        for k = 1:NumGVCAct %Run through all scenario 3s

            TotalDistances3(i,j,k) = Distance3InVC(i,j,k) + DistReq1(i) +
DistReq2(j);

        end

    end

end

DistReq3 = TotalDistances3;

end

function OpporValScenarios1 =
CalcOppValScenarios1(Products,Prox,ProductCompInd,GVCRCAScenarios1,GVCAct,B
aseRCASpace,ProxSums)

OpporVal1Time = tic;

NumGVCAct = size(GVCAct,1);
OpporValScenarios1 = zeros(NumGVCAct,1); %1) ContToComp (per activity)
Densities = zeros(size(Products,1),1);

ScenarioRCA = zeros(size(BaseRCASpace));

for Act = 1:NumGVCAct

    ScenarioRCA = GVCRCAScenarios1(Act,:); %+ transpose(BaseRCASpace);

    for i=1:size(Products,1) % Run through all potential products

        if ScenarioRCA(i) <= 1 % If opportunity not exploited yet
```

```
            DensityNumerator = 0;

            for j=1:size(Products,1) %Run through all columns of the
proximity matrix

                if ScenarioRCA(j) > 1 % If you are already competitive;
calculate Density from competitive to opportunity

                    if i ~= j % Make sure product is not not itself

                        DensityNumerator = DensityNumerator + Prox(i,j);

                    end

                end
            end

            Densities(i) = DensityNumerator / (ProxSums(i) - 1);

            %if RCA(i,2) <= 1 % If opportunity not exploited yet, add to
unexploited column

            OpporValScenarios1(Act) = OpporValScenarios1(Act) +
Densities(i) * ProductCompInd(i);

        end

    end

end

OpporVal1Timed = toc(OpporVal1Time)

end

function OpporValScenarios2 =
CalcOppValScenarios2(Products,Prox,ProductCompInd,GVCRCAScenarios2,GVCAct,B
aseRCASpace,ProxSums)

OpporVal2Time = tic;

%GVCRCAScenarios2 = zeros(NumGVCAct,NumGVCAct,NumProducts);

NumGVCAct = size(GVCAct,1);
OpporValScenarios2 = zeros(NumGVCAct,NumGVCAct,1); %1) Oppor Value (per
scenario)
Densities = zeros(size(Products,1),1);

ScenarioRCA = zeros(size(BaseRCASpace));

for Act1 = 1:NumGVCAct

    for Act2 = 1:NumGVCAct

        ScenarioRCA = GVCRCAScenarios2(Act1,Act2,:); %+
transpose(BaseRCASpace);
```

```
        for i=1:size(Products,1) % Run through all potential products

            if ScenarioRCA(i) <= 1 % If opportunity not exploited yet

                DensityNumerator = 0;

                for j=1:size(Products,1) %Run through all columns of the
proximity matrix

                    if ScenarioRCA(j) > 1 % If you are already competitive;
calculate Density from competitive to opportunity

                        if i ~= j % Make sure product is not not itself

                            DensityNumerator = DensityNumerator +
Prox(i,j);

                        end

                    end
                end

                Densities(i) = DensityNumerator / (ProxSums(i) - 1);

                %if RCA(i,2) <= 1 % If opportunity not exploited yet, add
to unexploited column

                OpporValScenarios2(Act1,Act2) =
OpporValScenarios2(Act1,Act2) + Densities(i) * ProductCompInd(i);

            end

        end

    end

end

OpporVal2Timed = toc(OpporVal2Time)

end

function OpporValScenarios3 =
CalcOppValScenarios3(Products,Prox,ProductCompInd,GVCRCAScenarios3,GVCAct,B
aseRCASpace,ProxSums)

%GVCRCAScenarios2 = zeros(NumGVCAct,NumGVCAct,NumProducts);

OpporVal3Time = tic;

NumGVCAct = size(GVCAct,1);
OpporValScenarios3 = zeros(NumGVCAct,NumGVCAct,NumGVCAct,1); %1) Oppor
Value (per scenario)
Densities = zeros(size(Products,1),1);

ScenarioRCA = zeros(size(BaseRCASpace));
```

```
for Act1 = 1:NumGVCAct

    for Act2 = 1:NumGVCAct

        for Act3 = 1:NumGVCAct

            ScenarioRCA = GVCRCAScenarios3(Act1,Act2,Act3,:); %+
transpose(BaseRCASpace);

            for i=1:size(Products,1) % Run through all potential products

                if ScenarioRCA(i) <= 1 % If opportunity not exploited yet

                    DensityNumerator = 0;

                    for j=1:size(Products,1) %Run through all columns of
the proximity matrix

                        if ScenarioRCA(j) > 1 % If you are already
competitive; calculate Density from competitive to opportunity

                            if i ~= j % Make sure product is not not itself

                                DensityNumerator = DensityNumerator +
Prox(i,j);

                            end

                        end
                    end

                    Densities(i) = DensityNumerator / (ProxSums(i) - 1);

                    %if RCA(i,2) <= 1 % If opportunity not exploited yet,
add to unexploited column

                    OpporValScenarios3(Act1,Act2,Act3) =
OpporValScenarios3(Act1,Act2,Act3) + Densities(i) * ProductCompInd(i);

                end

            end

        end

    end

    Act1

    Of = NumGVCAct

end

OpporVal3Timed = toc(OpporVal3Time)
```

```
end

function [BestFrom1Comp BestFrom1Oppor BestComplexityContr BestOpporGain] =
CalcBestFrom1(Round1Metrics,ScenariosDistances,BaseLineMetrics)

%Round1Metrics = [CompContr OpporGainScenarios1 DistReq1];

BestComplexityContr = zeros(size(ScenariosDistances,1),1);
BestOpporGain = zeros(size(ScenariosDistances,1),1);

BestFrom1Comp = zeros(size(ScenariosDistances,1),5); % 1) Activity1; 2)
Activity 2; 3) Activity 3; 4) BextComplexity Contr; 5) OpporGain; 6)
Distance
BestFrom1Oppor = zeros(size(ScenariosDistances,1),5); % 1) Activity1; 2)
Activity 2; 3) Activity 3; 4) Complexity Contr; 5) BestOpporGain; 6)
Distance

BestFrom1Comp(:,4) = BaseLineMetrics(2);

for i = 1:size(ScenariosDistances,1) %Repeat for all scenarios

    for j = 1:size(Round1Metrics,1) %Repeat for all activities

        if Round1Metrics(j,3) < ScenariosDistances(i) %Test distance
requirement

            if Round1Metrics(j,1) > BestComplexityContr(i) % Test if best
complexity for scenario

                BestComplexityContr(i) = Round1Metrics(j,1);

                BestFrom1Comp(i,1) = j; %Capture activity
                BestFrom1Comp(i,2) = 0;
                BestFrom1Comp(i,3) = 0;
                BestFrom1Comp(i,4) = Round1Metrics(j,1); %Capture
complexity contr
                BestFrom1Comp(i,5) = Round1Metrics(j,2); %Capture Oppor
gain
                BestFrom1Comp(i,6) = Round1Metrics(j,3); %Capture Distance

            end

            if Round1Metrics(j,2) > BestOpporGain(i); % Test if best Oppor
Gain for Scenario

                BestOpporGain(i) = Round1Metrics(j,2);

                BestFrom1Oppor(i,1) = j; %Capture activity
                BestFrom1Oppor(i,2) = 0;
                BestFrom1Oppor(i,3) = 0;
                BestFrom1Oppor(i,4) = Round1Metrics(j,1); %Capture
complexity contr
                BestFrom1Oppor(i,5) = Round1Metrics(j,2); %Capture Oppor
gain
                BestFrom1Oppor(i,6) = Round1Metrics(j,3); %Capture Distance

            end
```

```
        end

    end

end

end

function [BestFrom2Comp BestFrom2Oppor BestComplexityContr BestOpporGain] =
CalcBestFrom2(Comp2,DistReq2,OpporGainScenarios2,ScenariosDistances,BestFro
m1Comp,BestFrom1Oppor,BestComplexityContr,BestOpporGain)

%BestComplexityContr = zeros(size(ScenariosDistances,1),1);
%BestOpporGain = zeros(size(ScenariosDistances,1),1);

BestFrom2Comp = BestFrom1Comp; % 1) Activity1; 2) Activity 2; 3) Activity
3; 4) BextComplexity Contr; 5) OpporGain; 6) Distance
BestFrom2Oppor = BestFrom1Oppor; % 1) Activity1; 2) Activity 2; 3) Activity
3; 4) Complexity Contr; 5) BestOpporGain; 6) Distance

for i = 1:size(ScenariosDistances,1) %Repeat for all scenarios

    for j = 1:size(DistReq2,1) %Repeat for all activities

        for k = 1:size(DistReq2,1) %Repeat for all activities

            if DistReq2(j,k) < ScenariosDistances(i) %Test distance
requirement

                if Comp2(j,k) > BestComplexityContr(i) % Test if best
complexity for scenario

                    BestComplexityContr(i) = Comp2(j,k);

                    BestFrom2Comp(i,1) = j; %Capture activity 1
                    BestFrom2Comp(i,2) = k; %Capture activity 2
                    BestFrom2Comp(i,3) = 0;
                    BestFrom2Comp(i,4) = Comp2(j,k); %Capture complexity
contr
                    BestFrom2Comp(i,5) = OpporGainScenarios2(j,k); %Capture
Oppor gain
                    BestFrom2Comp(i,6) = DistReq2(j,k); %Capture Distance

                end

                if OpporGainScenarios2(j,k) > BestOpporGain(i); % Test if
best Oppor Gain for Scenario

                    BestOpporGain(i) = OpporGainScenarios2(j,k);

                    BestFrom2Oppor(i,1) = j; %Capture activity 1
                    BestFrom2Oppor(i,2) = k; %Capture activity 2
                    BestFrom2Oppor(i,3) = 0;
                    BestFrom2Oppor(i,4) = Comp2(j,k); %Capture complexity
contr
                    BestFrom2Oppor(i,5) = OpporGainScenarios2(j,k);
%Capture Oppor gain
                    BestFrom2Oppor(i,6) = DistReq2(j,k); %Capture Distance
```

```
                end

            end

        end

    end

end

end

function [BestFrom3Comp BestFrom3Oppor BestComplexityContr BestOpporGain] =
CalcBestFrom3(Comp3,DistReq3,OpporGainScenarios3,ScenariosDistances,BestFro
m2Comp,BestFrom2Oppor,BestComplexityContr,BestOpporGain)

%BestComplexityContr = zeros(size(ScenariosDistances,1),1);
%BestOpporGain = zeros(size(ScenariosDistances,1),1);

BestFrom3Comp = BestFrom2Comp; % 1) Activity1; 2) Activity 2; 3) Activity
3; 4) BextComplexity Contr; 5) OpporGain; 6) Distance
BestFrom3Oppor = BestFrom2Oppor; % 1) Activity1; 2) Activity 2; 3) Activity
3; 4) Complexity Contr; 5) BestOpporGain; 6) Distance

for i = 1:size(ScenariosDistances,1) %Repeat for all scenarios

    for j = 1:size(DistReq3,1) %Repeat for all activities

        for k = 1:size(DistReq3,1) %Repeat for all activities

            for m = 1:size(DistReq3,1) %Repeat for all activities

                if DistReq3(j,k,m) < ScenariosDistances(i) %Test distance
requirement

                    if Comp3(j,k,m) > BestComplexityContr(i) % Test if best
complexity for scenario

                        BestComplexityContr(i) = Comp3(j,k,m);

                        BestFrom3Comp(i,1) = j; %Capture activity 1
                        BestFrom3Comp(i,2) = k; %Capture activity 2
                        BestFrom3Comp(i,3) = m;
                        BestFrom3Comp(i,4) = Comp3(j,k,m); %Capture
complexity contr
                        BestFrom3Comp(i,5) = OpporGainScenarios3(j,k,m);
%Capture Oppor gain
                        BestFrom3Comp(i,6) = DistReq3(j,k,m); %Capture
Distance

                    end

                    if OpporGainScenarios3(j,k,m) > BestOpporGain(i); %
Test if best Oppor Gain for Scenario

                        BestOpporGain(i) = OpporGainScenarios3(j,k,m);
```

```
                        BestFrom3Oppor(i,1) = j; %Capture activity 1
                        BestFrom3Oppor(i,2) = k; %Capture activity 2
                        BestFrom3Oppor(i,3) = m;
                        BestFrom3Oppor(i,4) = Comp3(j,k,m); %Capture
complexity contr
                        BestFrom3Oppor(i,5) = OpporGainScenarios3(j,k,m);
%Capture Oppor gain
                        BestFrom3Oppor(i,6) = DistReq3(j,k,m); %Capture
Distance

                    end

                end

            end

        end

    end

end

end

function [MetricsForWinnersComp MetricsForWinnersOppor] =
CalcMetricsOfWinner(BestFrom3Comp,BestFrom3Oppor,GVCRCAScenarios3,GVCRCASce
narios2,GVCRCAScenarios1,Prox,ProxSums,Products,GVCFull)

%BestFrom3Comp = BestFrom2Comp; % 1) Activity1; 2) Activity 2; 3) Activity
3; 4) BextComplexity Contr; 5) OpporGain; 6) Distance
%BestFrom3Oppor = BestFrom2Oppor; % 1) Activity1; 2) Activity 2; 3)
Activity 3; 4) Complexity Contr; 5) BestOpporGain; 6) Distance

%GVCFullRCAScenarios3 = zeros(NumGVCAct,NumGVCAct,NumGVCAct,NumInGVCFull);
%GVCRCAScenarios3 = zeros(NumGVCAct,NumGVCAct,NumGVCAct,NumProducts);

%GVCFullRCAScenarios2 = zeros(NumGVCAct,NumGVCAct,NumInGVCFull);
%GVCRCAScenarios2 = zeros(NumGVCAct,NumGVCAct,NumProducts);

%GVCFullRCAScenarios1 = zeros(NumGVCAct,NumInGVCFull);
%GVCRCAScenarios1 = zeros(NumGVCAct,NumProducts);

NumScenarios = size(BestFrom3Comp,1);
NumProducts = size(GVCRCAScenarios1,2);

RCAOppor1 = zeros(NumProducts);
RCAOppor2 = zeros(NumProducts);
RCAOppor3 = zeros(NumProducts);
RCAComp1 = zeros(NumProducts);
RCAComp2 = zeros(NumProducts);
RCAComp3 = zeros(NumProducts);


MetricsForWinnersComp = zeros(NumScenarios,2);  % 1) Average distance to
products in the value chain with RCA < 1; 2) Average distance to products
in the products space with RCA <1
```

```
MetricsForWinnersOppor = zeros(NumScenarios,2); % 1) Average distance to
products in the value chain with RCA < 1; 2) Average distance to products
in the products space with RCA <1

for i = 1:NumScenarios % Run through all winners

    %% Part for Best From Comp

    CompA1 = BestFrom3Comp(i,1);
    CompA2 = BestFrom3Comp(i,2);
    CompA3 = BestFrom3Comp(i,3);

    if CompA1 ~= 0 %if true, there are 0 activities that meet the
specificiation.

        if CompA2 == 0 %If true, there is only 1 activity

            for j = 1:NumProducts

                RCAComp1(j) = GVCRCAScenarios1(CompA1,j);

            end

            MetricsForWinnersComp(i,1) =
CalcDistToVC(RCAComp1,GVCFull,Prox,ProxSums,Products);
            MetricsForWinnersComp(i,2) =
CalcDistToPS(RCAComp1,Prox,ProxSums,Products);

        else % If not true, then there are either 2 or 3 activities in
answer

            if CompA3 == 0 %If true, then there are 2 activities in answer

                for j = 1:NumProducts

                    RCAComp2(j) = GVCRCAScenarios2(CompA1,CompA2,j);

                end

                MetricsForWinnersComp(i,1) =
CalcDistToVC(RCAComp2,GVCFull,Prox,ProxSums,Products);
                MetricsForWinnersComp(i,2) =
CalcDistToPS(RCAComp2,Prox,ProxSums,Products);

            else %If not true, then there are 3 activities in answer

                for j = 1:NumProducts

                    RCAComp3(j) = GVCRCAScenarios3(CompA1,CompA2,CompA3,j);

                end

                MetricsForWinnersComp(i,1) =
CalcDistToVC(RCAComp3,GVCFull,Prox,ProxSums,Products);
                MetricsForWinnersComp(i,2) =
CalcDistToPS(RCAComp3,Prox,ProxSums,Products);
```

```
            end

        end

    end

    %% Part for Best From Oppor

    OpporA1 = BestFrom3Oppor(i,1);
    OpporA2 = BestFrom3Oppor(i,2);
    OpporA3 = BestFrom3Oppor(i,3);

    if OpporA1 ~= 0 %If  true, there are 0 activities

        if OpporA2 == 0 %If  true, there is only 1 activity

            for j = 1:NumProducts

                RCAOppor1(j) = GVCRCAScenarios1(OpporA1,j);

            end

            MetricsForWinnersOppor(i,1) =
CalcDistToVC(RCAOppor1,GVCFull,Prox,ProxSums,Products);
            MetricsForWinnersOppor(i,2) =
CalcDistToPS(RCAOppor1,Prox,ProxSums,Products);



        else % If not true, then there are either 2 or 3 activities in
answer

            if OpporA3 == 0 %If true, then there are 2 activities in answer

                for j = 1:NumProducts

                    RCAOppor2(j) = GVCRCAScenarios2(OpporA1,OpporA2,j);

                end

                MetricsForWinnersOppor(i,1) =
CalcDistToVC(RCAOppor2,GVCFull,Prox,ProxSums,Products);
                MetricsForWinnersOppor(i,2) =
CalcDistToPS(RCAOppor2,Prox,ProxSums,Products);

            else %If not true, then there are 3 activities in answer

                for j = 1:NumProducts

                    RCAOppor3(j) =
GVCRCAScenarios3(OpporA1,OpporA2,OpporA3,j);

                end

                MetricsForWinnersOppor(i,1) =
CalcDistToVC(RCAOppor3,GVCFull,Prox,ProxSums,Products);
```

```
                MetricsForWinnersOppor(i,2) =
CalcDistToPS(RCAOppor3,Prox,ProxSums,Products);

            end

        end

    end

end

end

function AvgDistToVCNotAqYet =
CalcDistToVC(ProdRCA,GVCFull,Prox,ProxSums,Products)

%GVCFull 1) Tier#; 2) GVC Activity#; 3) HS Code; 4 Complexity; 5) Density;
6) RCA (7) Complexity if opp; 8) Distance if Opportunity; 9) OpporGain if
Opportunity; 10)RCA if Opp; 11) SA Export of Activity; 12) World Export of
activity; 13) SA export if oppor; 14) World export if opp; 15) ProdPosition

%% Calculate Distance To Products In The Product Space Not yet Acquired

NumberNotAcquired = 0;
Distance = zeros(size(Products,1),1);
DistanceNumerator = zeros(size(Products,1),1);

for i=1:size(GVCFull,1) % Run through all potential products

    if ProdRCA(GVCFull(i,15)) <= 1 % If opportunity not exploited yet,
distance to it can be calculated.

        NumberNotAcquired = NumberNotAcquired + 1;

        for j=1:size(Products,1) %Run through all columns of the proximity
matrix

            if ProdRCA(j) < 1 % If you are not yet competitive; add to
distance to opportunity

                if i ~= j % Make sure product is not not itself

                    DistanceNumerator(i) = DistanceNumerator(i) +
Prox(GVCFull(i,15),j);

                end

            end
        end

        Distance(i) = DistanceNumerator(i) / (ProxSums(GVCFull(i,15)) - 1);

    end

end

SumDistances = sum(Distance);
```

```
AvgDistToVCNotAqYet = SumDistances / NumberNotAcquired;

end

function AvgDistToPSNotAqYet = CalcDistToPS(ProdRCA,Prox,ProxSums,Products)

%ProdRCA
%GVCFullRCA

%% Calculate Distance To Products In The Product Space Not yet Acquired

NumberNotAcquired = 0;
Distance = zeros(size(Products,1),1);
DistanceNumerator = zeros(size(Products,1),1);

for i=1:size(Products,1) % Run through all potential products

    if ProdRCA(i) <= 1 % If opportunity not exploited yet, distance to it
can be calculated.

        NumberNotAcquired = NumberNotAcquired + 1;

        for j=1:size(Products,1) %Run through all columns of the proximity
matrix

            if ProdRCA(j) < 1 % If you are not yet competitive; add to
distance to opportunity

                if i ~= j % Make sure product is not not itself

                    DistanceNumerator(i) = DistanceNumerator(i) +
Prox(i,j);

                end

            end
        end

        Distance(i) = DistanceNumerator(i) / (ProxSums(i) - 1);

    end

end

SumDistances = sum(Distance);
AvgDistToPSNotAqYet = SumDistances / NumberNotAcquired;


end
```