**Report of Changes 2018/Oct/12**

1. The way the average is calculated is changed.

First in Person state *yearly\_interactions* we reverse the called of methods, we call first the methods for *Set\_LastYear\_Average*\_ (OGS, IGS,ShNr; and *this\_year\_average* is reset to 0). Then we set the *cumulative\_average* to the *last\_year\_average*. In other words, the cumulative average is not the cumulative over all years an agent has lived but the one of the past year only.

Consequently, the method *SetCumulativeAveragesOutInSharedNorms* was changed, this cumulative average corresponds now to the one of the past year and not of overall years of the simulation.

1. We did the same for the links and interaction ratios Maj:Min.

Two agents’ variables added: last year\_average\_Maj\_Min\_interactions(links)\_ratio.

Two methods added: Set\_LastYear\_Average\_Maj\_Min\_interactions(links)\_ratio.

Methods called in the yearly\_updates state together with the methods for OGS,IGS and ShNo

SetCumulativeAveragesMajMinRatios changed so the current average is set to past year.

1. All variables in Main related to the cumulative averages were added a prefix “Cum\_”
2. The names of the two methods collecting cum averages were changed, *Cum* was added
3. A new variable was added in main: “Employment availability” calculated in the weekly\_list\_updates even in main as:

**double** Jobs\_available = 0;

**double** Jobs\_Taken = 0;

**for** (**int** i = 0; i < employment\_Places.size(); i++)

{

Jobs\_available += (**double**) employment\_Places.get(i).Number\_Of\_Jobs;

Jobs\_Taken += (**double**) employment\_Places.get(i).Current\_Employed;

}

Employment\_ availability = (Jobs\_available - Jobs\_Taken) / Jobs\_available;

Employment\_ availability then replaces the parameter Employment\_Change\_Pct\_Majority when individuals are NOT\_EMPLOYED and looking for a job in the method “Update\_Montly\_Employment” in Person.

1. Number\_of\_Employers added to the initial number of agents for Agent Employment\_Place in Main.
2. A modification was done to the algorithm determining the chances of the minority employed of losing their job. Previously it was:

else if ((…)&&(chance\_of\_employmentnotChange < (get\_Main().Employment\_Change\_Pct\_Majority \* get\_Main().Enforced\_Antidiscr)))

Because Enforced\_Antidiscrimination takes values [0,1], the chance of the minority losing their job were <= than that of the MAJORITY (chance\_of\_employmentnotChange). To avoid this and make the chances of the MINORITY losing their job higher than that of the MAJORITY the following change was introduced:

**else** **if** ((…)&&(chance\_of\_employmentnotChange < ( (get\_Main().Employment\_Change\_Pct\_Majority)\*(1+(1-get\_Main().Enforced\_Antidi))

In this way the chances of the MINORITY losing their job are higher than those of MAJORITY, and the increase is inversely proportional to the value of Enforced\_Antidiscrimination value. So, if enforced antidiscrimination is 0.9, 0.6, 0.3, 0.2, the increase is 0.1, 0.4, 0.7, and 0.8 respectively.

**Report of Changes and additions 2018/Oct/19**

1. In Person, function Set\_Outgroup\_Suspicion. Standard deviations of the values for the maj/min groups were wrongly set. Now they are correct.
2. A new output file was added. This file captures individuals’ attributes annually and includes all agents. This should be considered when doing analyses, since agents <= 12 do not start interactions yet and their IGS, OGS and ShNr values are averages of that of their parents values.
3. A new parameter SimulationID was added in Main. This parameter keeps track of the Iteration number and replication number. The default value is set to “NA”. If needed, it has to be set in the “Before simulation run” field in a given experiment (e.g. see Monte Carlo experiment).

**Changes made to CRED 2018/Oct/30**

Methods added: Set\_ThisYear\_Average\_WorldView (Frustration), Set\_Last year\_Average\_WorldView (Frustration), Update\_Kids\_WorldView (NOTE, should frustration also be set to the average of parents)

**Changes made to SIMLAN 2019/Oct/7**

1) Histograms for age at marriage, Age distribution, Education, and Income added.

2) Histogram data added to keep values of demographics.

3) Update of age at marriage histogram is done in set\_family function in person tab.

**if**(**this**.Flag == **false**)

{

get\_Main().Hist\_Mal\_Maj\_Marriage.add(**this**.Age);

get\_Main().Hist\_Fem\_Maj\_Marriage.add(SelectedFemale.Age);

}…

**if**(**this**.Flag == **false**)

{

get\_Main().Hist\_Mal\_Min\_Marriage.add(**this**.Age);

get\_Main().Hist\_Fem\_Min\_Marriage.add(SelectedFemale.Age);

}…

**if**(**this**.Flag == **false**) {

**if**(**this**.Group == MAJORITY\_GROUP){ get\_Main().Hist\_Mal\_Maj\_Marriage.add(**this**.Age); }

**if**(**this**.Group == MINORITY\_GROUP){ get\_Main().Hist\_Mal\_Min\_Marriage.add(**this**.Age); }

**if**(selectedFemale.Group == MAJORITY\_GROUP){ get\_Main().Hist\_Fem\_Maj\_Marriage.add(selectedFemale.Age); }

**if**(selectedFemale.Group == MINORITY\_GROUP){ get\_Main().Hist\_Fem\_Min\_Marriage.add(selectedFemale.Age); }

}

4) Update of education histogram when total education is set in set education function, person tab

*get\_Main().Hist\_Edu\_Min.add(p.Education\_Total);*

*…*

*get\_Main().Hist\_Edu\_Min.add(p.Education\_Total);*

5) Update of income and age histograms, in *initial\_Pop\_Histogram* event and in the function *Update\_Pop\_Hist*, both in main. The function *Update\_Pop\_Hist* is called in *Annual\_updates*

*Hist\_Age\_Maj.reset();*

*Hist\_Age\_Min.reset();*

*List <Person> Maj = findAll(People, p -> p.Group == MAJORITY\_GROUP);*

*List <Person> Min = findAll(People, p -> p.Group == MINORITY\_GROUP);*

***for*** *(Person p : Maj) { Hist\_Age\_Maj.add(p.Age); }*

***for*** *(Person p : Min) { Hist\_Age\_Min.add(p.Age); }*

*Hist\_Income\_Maj.reset();*

*Hist\_Income\_Min.reset();*

*List <Person> Maj\_Emp = findAll(People, p -> p.Group == MAJORITY\_GROUP && p.Employment != STUDENT);*

*List <Person> Min\_Emp = findAll(People, p -> p.Group == MINORITY\_GROUP && p.Employment != STUDENT);*

***for*** *(Person p : Maj\_Emp) { Hist\_Income\_Maj.add(p.Income\_Individual); }*

***for*** *(Person p : Min\_Emp) { Hist\_Income\_Min.add(p.Income\_Individual); }*

The distribution of age of marriage follows a poisson truncated distribution. Age\_of\_Marriage variable added to person. This variable is set in a new function Set\_Marriage\_Age and this function is called in the Generate\_Children state.

The distribution of age of initial populations follow a custom distribution (one for adults and one for kids) and for each majority, minority group. The age of the initial adult population is set in set age majority and set age minority function in person tab. The age of the initial offspring population (<24) is set in set children with the following code:

***else******if*** *(****this****.Flag==****true****)*

*{*

*MyChild.Age = MyWife.get(0).Age;*

***while****((MyWife.get(0).Age-MyChild.Age) < 15 | (MyWife.get(0).Age-MyChild.Age) > 60)*

*{*

***if****(****this****.Group==MAJORITY\_GROUP){MyChild.Age=get\_Main().AgeDistKidsMaj.getInt();}*

***if****(****this****.Group==MINORITY\_GROUP){MyChild.Age=get\_Main().AgeDistKidsMin.getInt();}*

*}*

Agents of initial population starting married, set all children at once, mothers > 60 years old cannot have child though and the age difference between mother and offspring is at least 15 years. Agents of generation 1 and above with over 60 years old cannot have children anymore. The Features of new children are set in a new function Set\_Kid\_Features. Code in Set children added:

*/// Agents from initial population that started being married*

***if*** *(****this****.Flag==****true****)*

*{*

*// Generate all children at once*

***for****(****int*** *i=0; i <* ***this****.Total\_Number\_Of\_Children; i++)*

*{*

*Person MyChild = get\_Main().add\_People();*

***this****.Children.connectTo(MyChild);*

***this****.My\_Children += 1;*

*MyWife.get(0).Children.connectTo(MyChild);*

*MyWife.get(0).My\_Children += 1;*

*MyChild.Age = MyWife.get(0).Age;*

***while*** *( (MyWife.get(0).Age - MyChild.Age) < 15 | (MyWife.get(0).Age - MyChild.Age) > 60)*

*{*

***if****(****this****.Group == MAJORITY\_GROUP){ MyChild.Age = get\_Main().AgeDistKidsMaj.getInt(); }*

***if****(****this****.Group == MINORITY\_GROUP){ MyChild.Age = get\_Main().AgeDistKidsMin.getInt(); }*

*}*

*MyChild.Set\_Kid\_Features(MyWife.get(0),* ***this****);*

*//if (MyWife.get(0).Generation == 1) { traceln("Age difference: " + (MyWife.get(0).Age - MyChild.Age) ); }*

*//Inform connections of Birth*

***this****.send\_birth\_notifications( MyChild );*

*}*

*}*

*/// Agents from initial population that were single at arrival or agents from generation 2 and above*

***if****(* ***this****.Flag ==* ***false*** *&& chance\_of\_having\_children < MyWife.get(0).Age && MyWife.get(0).Age < 60)*

*{*

*Person MyChild = get\_Main().add\_People();*

***this****.Children.connectTo(MyChild);*

***this****.My\_Children += 1;*

*MyWife.get(0).Children.connectTo(MyChild);*

*MyWife.get(0).My\_Children += 1;*

*MyChild.Age=0;*

*MyChild.Set\_Kid\_Features(MyWife.get(0),* ***this****);*

*//traceln("Mom Age: " + MyWife.get(0).Age + " My Kid Age " + MyChild.Age);*

*//Inform connections of Birth*

***this****.send\_birth\_notifications( MyChild );*

*}*

Function Set children changed so they don’t have one child more than the total number assigned. Code changed from

***if*** *(…****this****.My\_Children* ***<=******this****.Total\_Number\_Of\_Children )*

to

***if*** *(…****this****.My\_Children* ***<******this****.Total\_Number\_Of\_Children )*

Update\_Monthly\_Employment function modified so the percentage of employment set at initialization are kept during the whole simulation. NOTE: the actual percentage of maj/min mal/fem agents employed is equal to the percentage parameter – Employment\_Change\_Pct+Majority. So if males majority employment percentage is 100% and employment change = 5%, the actual percentage of males from majority employed is 95%. To keep these percentages constant, the chance of obtaining a job when unemployed was calculated as:

(PE – Change) \* Change / (1 – (PE -Change) )

Where PE is initial parameter percentage of maj/min fem/mal employed. So, the Pseudocode, calculated in main in *store\_init\_stats* state and weekly\_updates events is:

MAJORITY

*Emp\_gain\_mal\_maj = ((Per\_Maj\_Male\_Emp - Emp\_Ch\_Pct\_Maj) \* Emp\_Ch\_Pct\_Maj )/ ( 1 - ((Per\_Maj\_Male\_Emp - Emp\_Ch\_Pct\_Maj)));*

*{Emp\_gain\_fem\_maj = ( (Per\_Maj\_Fem\_Emp - Emp\_Ch\_Pct\_Maj) \* Emp\_Ch\_Pct\_Maj)/ ( 1 - ((Per\_Maj\_Fem\_Emp - Emp\_Ch\_Pct\_Maj)));*

MINORITY

*Emp\_gain\_mal\_min = ((Per\_Min\_Male\_Emp - Emp\_Ch\_Pct\_Maj) \* Emp\_Ch\_Pct\_Min )/ ( 1 - ((Per\_Min\_Male\_Emp - Emp\_Ch\_Pct\_Min)));*

*Emp\_gain\_fem\_min = ( (Per\_Min\_Fem\_Emp - Emp\_Ch\_Pct\_Maj) \* Emp\_Ch\_Pct\_Min)/ ( 1 - ((Per\_Min\_Fem\_Emp - Emp\_Ch\_Pct\_Min)));*

Then in Update\_Monthly\_Employment the variable employment\_change\_gain is adjusted accordingly.

***if****(****this****.Gender == MALE) { Employment\_Change\_gain = get\_Main().Emp\_Gain\_Mal\_Maj;}*

***if****(****this****.Gender == FEMALE) {Employment\_Change\_gain = get\_Main().Emp\_Gain\_Fem\_Maj;}*

***if****(****this****.Gender == MALE) { Employment\_Change\_gain = get\_Main().Emp\_Gain\_Mal\_Maj;}*

***if****(****this****.Gender == FEMALE) {Employment\_Change\_gain = get\_Main().Emp\_Gain\_Fem\_Maj;}*

Further, to avoid unemployment due to job unavailability, when companies run out of jobs, then one company is selected at random and a job position is added. This is done in the function set\_employment\_network

*List <Employment\_Place> Employers = findAll(get\_Main().employment\_Places,p -> (p.Current\_Employed < p.Number\_Of\_Jobs) );*

*/// Employers with job positions*

***if*** *( (Employers.size() != 0) && (found\_job ==* ***false****) )*

*{*

*job\_id = uniform\_discr(Employers.size() - 1);*

***this****.Employer = Employers.get(job\_id);*

*Employer.Current\_Employed++;*

*found\_job =* ***true****;*

*}*

*/// Employers without job positions, select one at random and create one extra job*

***else***

*{*

*Employment\_Place NewEmployer = randomFrom(get\_Main().employment\_Places);*

***this****.Employer = NewEmployer;*

*NewEmployer.Current\_Employed++;*

*NewEmployer.Number\_Of\_Jobs++;*

*found\_job =* ***true****;*

*}*

The function set\_Income is now called also inside SET\_EMPLOYMENT, it was out commented in state GENERATE\_CHILDREN and in function Set\_Maturation\_Process. In the function update\_Monthly\_Employment the set of income was changed. When the state of the agent goes from unemployed to employed, one checks if the agents was already employed and if so, the previous income is once again assigned, otherwise a first income is set. Code:

*/// Has already been employed set income to last income when employed*

***if****(****this****.Income\_When\_Last\_Employed > (Income\_Min\_In\_Hundred\_Thousands \* 100000))*

*{*

***this****.Income\_Individual = Income\_When\_Last\_Employed;*

*}*

*// Has never been employed set first income*

***else***

*{*

***this****.Set\_Income();*

*}*

/// COUNTING INTERGROUP INTERACTIONS

Added 2 new variables to agents: intergroup interactions positive/negative. These variables are counters updated in the following functions: 1) Set Impersonal Interaction; 2) Set Minority Majority Ego Interaction; 3) Set Minority Neigborhood Interaction; 4) Set OFSN Interaction; 5) Set OSN Interaction. In all these functions counters of interactions Min-Min and Maj-Maj were added because although these interactions have no effect on IGS, OGS and ShNr, the interactions should be counted to get the precise ratio of interactions Maj/Maj+Min.

Further, 4 other variables added: This Year Average Pos Neg Intergroup Ratio, Last Year Average Pos Neg Intergroup Ratio, This Year Average Pos Neg Samegroup Ratio, Last Year Average Pos Neg Samegroup Ratio. These variables keep the ratio of Pos/Pos+Neg every year. This Year… variable are updated weekly with function *Set\_ThisYear\_Pos\_Neg\_Intergroup\_Interactions\_Ratio()*; and *Set\_ThisYear\_Pos\_Neg\_Samegroup\_Interactions\_Ratio()*; and the Last Year… variable are updated in function S*et\_LastYear\_Average\_Pos\_Neg\_Intergroup\_Interactions\_Ratio();* S*et\_LastYear\_Average\_Pos\_Neg\_Samegroup\_Interactions\_Ratio();*

For the cumulative average another variable was added *AveragePos\_Neg\_Intergroup\_Interactions\_Ratio*

And *AveragePos\_Neg\_Samegroup\_Interactions\_Ratio* this variable is updated yearly in the function *SetCumulativeAverageMajMinRatios.*

In main, two new variables added Cum\_AverageMajorityRatioNegPosInteractions and one for the minority respectively and AverageMinorityRatioNegPosSameGroupInteractions. These variables are updated in GetPopCumRatiosMajMin

Output files: Update\_AnnualCSV\_CustomExp, Update\_CSV, Update\_CSV\_CustomExp, Update\_MonteCarlo\_CSV, Update\_Annual\_Individual\_CSV\_CustomExp