User's Guide for the Stochastic Conflict Model

By Scott D. Peckham

Introduction

The Stochastic Conflict Model was developed for Kimetrica/AIR, with funding from the DARPA World Modelers program. It is an open-source model, written entirely in Python, and distributed on GitHub as a Python package. The package includes the model itself, a set of utilities, documentation, a graphical user interface (GUI), and a Jupyter notebook that shows how to use the model and its GUI. While there are several ways to run the model as explained in the section: Different Ways to Run the Model, the GUI is the simplest.

While the model was written so that it can be used for any region on Earth (i.e., for any geographic bounding box), it has so far only been applied to a region that spans the Greater Horn of Africa. The phrase "Horn of Africa" refers to the countries: Djibouti, Eritrea, Ethiopia, Somalia (and the de facto country, Somaliland). The phrase "Greater Horn of Africa" additionally includes the countries: Kenya, Uganda, Sudan, and South Sudan. The region spanned by the model (using defaults) fully contains the Greater Horn of Africa countries, as well as some piece of the following countries: Congo, Tanzania, Rwanda, Burundi, Yemen, Saudi Arabia, Egypt, and others. The section titled Examples for the Greater Horn of Africa contains many images generated using different parameter settings.

This user's guide consists of the following sections:

Conceptual Overview of the Model

The Stochastic Conflict Model is loosely based on a cellular automaton model known as the "Forest-fire model" (Drossel and Schwabl , 1992; Forest-fire Model, 2021) which is a very simple model for how forest fires emerge and spread. Conflicts have some similarities to forest fires in that they are ***triggered*** by random events, they ***spread*** spatially to other grid cells, and they are eventually **resolved**. Due to these broad similarities, the Forest-fire model served as the inspiration for the current model, even though that model is much simpler. A brief review of the Forest-fire model is given in Appendix 1. Also see papers listed in the section References – Forest-Fire Model.

The Stochastic Conflict Model is ***stochastic*** (i.e., it uses probability theory and random variables), ***dynamic*** (i.e., the model state evolves in time), ***geospatial*** (i.e., the model domain is a grid), and ***data-driven***. It is also ***mechanistic***, in the sense that it explicitly simulates the important processes of conflict emergence, spreading and resolution. (Note that machine learning models do not represent these mechanisms.) Its main purpose is to help the modeler understand the mechanics of conflict and how each of these processes and various indicators influence the dynamic character of a conflict. The model can exhibit a very wide range of behaviors, depending on the values of its input parameters and input grids. It consists of the following 4 key components.

1. Unrest - The Potential for Conflict to Emerge. For each geospatial grid cell, we define a variable called ***unrest***, which is intended to quantify the potential for conflict within that grid cell. This is a composite variable or ***index*** that is a function of one or more ***indicator*** variables. Here, we define an ***indicator*** to be any variable that may help to quantify a *qualitative property* of an object or system, but that by itself provides only an indirect or insufficient measure of that property. There are many indicator variables that can be used (and have been used) in an effort to quantify the potential for conflict and to predict the emergence of conflict. These can be roughly grouped into the following categories.

(1) **climatic conditions** (e.g. rainfall amount, drought, flooding, heat-wave, sea-level rise)
(2) **economic conditions** (e.g. food shortage, food prices, unemployment, income disparity, resource richness, inflation rate, GDP/capita, crop failure),
(3) **social conditions** (e.g. population count, population density, population growth, quality of life, standard of living, infant mortality rate, "identity group" mixture, education, access to health care, change in ethnic mix, maybe due to migration, land cover change, deforestation, "youth bulge", disasters, urbanization)
(4) **geographic conditions** (e.g. distance to border, distance to capital, remoteness)
(5) **political conditions** (e.g. time to next election, polarization, regime change, assasination of a leader, ethnic cleansing)
(6) **historical conditions** (e.g. time since the last conflict)

Note: We do not expect for all of these variables to be equally important in determining the degree of unrest. Previous modeling work in Ethiopia used a machine-learning algorithm and

found that population density, the amount of rainfall, and land cover change were among the most important indicators. Note that these 3 indicators may be less dominant in other countries.

Note: Machine-learning algorithms/models are one method than can be used to create a grid of "unrest" values for a given region from various indicators.

Note: In the forest-fire analogy, unrest might correspond to the degree of "flammability", "combustibility", "dryness" or "fuel availability" (e.g. biomass). In the forest fire model, however, "flammability" is a boolean property -- it is 1 if there is a tree in the grid cell and 0 if there isn't. See Combustibility and Flammability (2021).

2. Trigger Events - The Initiation of Conflict. For each grid cell, there is a probability that some event will occur that initiates or sparks a conflict. (And that initial conflict may escalate or de-escalate, depending largely on how it is handled and the power discrepancy between the parties.) Conflicts are initiated as a result of negative interactions between people. This simple fact helps to explain the strong correlation between the occurrence of conflicts and population count or density. Let p(i) denote the probability that a conflict is initiated in a grid cell with index i. We then expect that p(i) will be an increasing function of --- and likely proportional to --- U(i), the degree of unrest in the ith grid cell.

Note: This also differs from the forest-fire model of Drossel and Schwabl (1992). In that model, every grid cell has the same probability of a trigger event such as a lightning strike. In this model, we expect this probability to vary spatially (and also in time) and to depend on the degree of unrest. (Of course, unquenched campfires and discarded cigarettes can also trigger a forest fire, but these are not naturally-occurring events. Since they are caused by people, the probability of these types of trigger events would increase with the population count of a grid cell.)

3. Connectivity - The Potential for Conflict to Spread. For each geospatial grid cell, we also compute a measure of connectivity that is used to quantify the potential for a conflict in one cell to spread to other grid cells. This is also a composite variable or index that is a function of several indicator variables. The indicator variables can characterize two distinct types of connectivity:

(1) **Local / Geographic Connectivity** (geospatial or physical; e.g. transportation network accessibility, road density, topographic roughness, travel times)

(2) **Nonlocal / Electronic Connectivity** (access to nonlocal information; e.g. access to devices: cell phone, TV, radio, internet/broadband. Also social media usage, newspapers.

The widespread use of cell phones and social media has led to a dramatic increase in "electronic connectivity". It is often viewed as the key factor in the "Arab Spring" phenomenon. See Arab Spring (2021). Note that electronic connectivity may have either a positive or negative impact on the spreading (e.g. when the family of someone who was killed calls for peaceful demonstrations, conflict may be reduced).

Note: It appears that the ***communication*** between people in different grid cells (near or far) is the primary mechanism by which conflict spreads. So connectivity is really getting at the ability to communicate between cells.

Note: The standard forest-fire model includes only local (nearest neighbor) connectivity, and does not allow any type of nonlocal connectivity.

Note: While TVs, radios and newspapers allow people to ***receive*** information, they are much less likely to be used by people to ***send*** information to others. This is in sharp contrast with telephones (cell or land-line), and internet access which allow two-way communication. In order for information to spread from one cell to another, it has to first be received by some means and then sent or forwarded via two-way communication. Perhaps we should therefore distinguish between the degree to which people in a grid cell ***receive*** information electronically (by TV, radio, newspaper, phone, internet), and the degree to which they ***send*** it electronically (only by phone & internet). The time it takes for a person to act on information received (e.g. forward it, organize a protest, write about it) should probably also be taken into account.

4. Conflict Duration - The Persistence of Conflict. Conflicts eventually get resolved, and therefore have a finite (but possibly long) duration. This can be modeled in various ways. One of the simplest is to treat the duration of each conflict event as an independent, random variable drawn from some distribution that takes only positive values. For example, random variables from a geometric distribution (Geometric Distribution, 2021) are positive integers and could be used to determine duration as one or more discrete time intervals. Lee et al. (2020b) discuss conflict duration in terms of "virulence".

In the current version of the model, we generate a Bernoulli random variable (parameter p) in each time interval to determine whether the conflict in a grid cell ends in that time interval. Bernoulli random variables (Bernoulli Distribution, 2021) have only 2 possible values: 1 with probability, p, and 0 with probability, (1-p). As is well-known in probability theory, it follows that the conflict durations then have a Geometric distribution with the same parameter, p.

Mathematical Overview of the Model

In this section, we explain how the conceptual model outlined in the previous section has been defined mathematically. While this section clarifies how the model's input parameters are used, some readers may wish to skip to the next section.

Let U denote a geospatial grid of values that quantify unrest, which will typically be computed from several indicators. Let C1 denote a geospatial grid of values that quantify the average degree to which people in a given grid cell ***communicate with*** people in nearest neighbor (adjacent) grid cells. Let C2 denote a geospatial grid of values that quantify the average degree to which people in a grid cell communicate with (i.e. send or receive information to/from) people in any other grid cell. In general, U, C1, and C2 values are expected to vary both spatially and in

time. (These may be estimated by a machine learning algorithm or some other algorithm as functions of selected indicator variables.)

Note: C2 depends on the degree to which a given cell is connected electronically to the rest of the country (or region or world).

Conflict Emergence

Let E(i,j,k) denote the event that a conflict emerges during the kth discrete time interval, in the grid cell that has column and row indices i and j. Let p(i,j,k) be the probability of the event E(i,j,k):

$$p(i,j,k) = p\_emerge(i,j,k) = Prob[\ E(i,\ j,\ k)]$$

We assume that p(i,j,k) is an increasing function of U(i,j,k). The current version of the model further assumes that p(i,j,k) is ***proportional*** to U(i,j,k), so that:

$$\text{if } (S(i,j,k) = 0), \quad p(i,j,k) = c\_emerge * U(i,j,k) / max\_i,j\ [\ U(i,j,k)\ ]$$
$$\text{if } (S(i,j,k) = 1), \quad p(i,j,k) = 0,$$

where c_emerge is in (0,1]. Here, we have normalized the grid U(i,j,k) and then multiplied the result by a constant, c. Dividing by the array max and multiplying by a model parameter c in (0,1] guarantees that 0 <= p(i,j,k) < 1 for every grid cell --- as it must since it is a probability. Note that the parameter c can be adjusted to make conflicts more or less likely to occur. In the special case where all values in the U grid are equal, all cells will have the same probability of conflict emergence: p_emerge = c_emerge. (Note: Instead of dividing by the array max, we could have divided by the array sum, but then the upper bound on c is more complicated.)

Let S be a geospatial grid with the same dimensions as U, C1 and C2, such that the value of a grid cell in S is 1 if the cell is experiencing (or engaged in) a conflict, and 0 otherwise. For each model timestep, and for each grid cell, generate an ***independent***, random variable from a Bernoulli distribution (see Bernoulli Distribution, 2021) such that:

$$S(i,j,k) = 1 \text{ with probability } p(i,j,k) \text{ and}$$
$$S(i,j,k) = 0 \text{ with probability } (1 - p(i,j,k)).$$

Recall that the E[S(i,j,k)]= p(i,j,k) for a Bernoulli random variable. Also recall that the expected value of a sum of random variables is equal to the sum of their expected values, *even if they are not independent*. Therefore, the expected number of grid cells in which a conflict is ***initiated*** in the 1st time interval (k=1, before any spreading) is given by:

$$E(\ Sum\_i,j\ [S(i,j,1)] = (\ \text{total conflicts in 1st interval}\ ) = Sum\_i,j\ [p(i,j,1)]$$

The expected number of grid cells in which a conflict is initiated in the kth time interval (k >1, before spreading), will be less than this since cells already experiencing conflict (i.e. with S=1) are excluded (i.e. for them, p(i,j,k) = 0).

Conflict Spreading

For every grid cell that has S(i,j,k) = 1, the next step is to determine if conflict spreads to other grid cells (which could be nearest neighbors or far away due to electronic connectivity). If it does, the spreading process is repeated for those other grid cells until there is no further spreading. (That is, no cells have their S value changed to 1.) Since there is a lag time between cell-to-cell communication and a resulting action (to spread conflict into a new cell), we introduce a time lag parameter, tau, which can be taken to be a multiple of the model timestep. (In the current version of the model, we assume that tau = 1.)

There are 4 special cases of spreading that can be considered:
Case 1: no spreading,
Case 2: spreading only by local connectivity,
Case 3: spreading by only nonlocal connectivity, and
Case 4: spreading by both local and nonlocal connectivity.
For the last three of these cases, various spreading rules are also possible.

Case 1. No spreading. While there may be conflict in many cells, there is no spreading of conflict to other grid cells.

Case 2. Spreading only by local connectivity. In this case, conflict can only spread from a grid cell with conflict to other grid cells that have sufficiently high C1 values. Cells with a high C1 value should include nearest neighbor grid cells and possibly other cells, such as those separated by a very short travel time. In the current version of the model, the probability of spreading to any ***nearest neighbor*** is weighted by the C1 value of that neighbor, i.e.

$$p(n) = c\_spread * C1(n) / max\_n(C1(n)),$$

where c_spread is in (0,1] and n is a nearest neighbor index. In the special case where all values in the C1 grid are equal, all cells will have the same probability of conflict spreading: p_spread = c_spread. Conflict is not allowed to spread to a neighbor that has U=0, and this includes grid cells in a lake or in the ocean. In the current version, conflict cannot "spread to" or "re-emerge" in a cell that already has conflict, but this could be changed. Various other spreading rules are also possible, such as:

(a) Conflict will spread to any neighbor, n, such that U(n) > U0, where U0 is a threshold value.
(b) Conflict will spread to any neighbor, n, such that U(n) >= a * U(i), with a in (0,1].
(c) The probability of spreading to any neighbor is weighted by the U and C1 values of that neighbor, for example using:

$$p(n) = c\_spread * [U(n) * C1(n)] / max\_n [U(n) * C1(n)],$$

where c_spread is in (0,1]. Here, the probability of spreading to a neighbor is low if either U(n) or C1(n) is low.

Case 3. Spreading only by nonlocal connectivity. In this case, conflict can spread from a grid cell with conflict to any other grid cells that have sufficiently high C2 and/or U values.

Case 4. Spreading by both local and nonlocal connectivity. In this case, conflict can spread from a cell with conflict to:

(1a) any of its nearest neighbor grid cells if its own C1 value is sufficiently high, and if their value of U is sufficiently high, OR
(1b) any of its nearest neighbor grid cells with sufficiently high C1 and U values.

(2) any other grid cell with sufficiently high C2 and U values.

Conflict Resolution and Duration

For every grid cell that had its S value set to 1 in the kth time interval (by emergence/triggering or spreading), generate an independent random variable from a Bernoulli distribution (Bernoulli Distribution, 2021) with parameter, p_resolve. If B=1, the conflict is ***resolved*** in time interval k and if B=0 the conflict continues. If we take X to be the random number of time intervals before the conflict is resolved, which is the duration of the conflict, then X will have a Geometric distribution (Geometric Distribution, 2021) with parameter, p_resolve. The expected value of this random duration is then equal to 1/p_resolve. (So a smaller value of p_resolve results in a larger mean duration.) As long as conflict is active within a cell (one or more timesteps) it will attempt to spread to other cells.

Explanation of Key Model Parameters

The following parameters can be changed in the model's configuration file, using a text editor, or in the model's GUI. Here, the "long name" (in bold) is followed by the "key name" (in parentheses) that is used in the config file.

Number of timesteps (n_steps): The number of timesteps for which to run the model. The duration of a single timestep is determined by the time_units parameter. Model runtime is proportional to n_steps. (Type: Integer, Default value: 100)

Unrest grid file (U_file): A spatial grid of values (2D array) that gives a measure of the "unrest", or potential for conflict to emerge, in each grid cell. It can depend on indicators such as population count, average rainfall rate, and many others. It is fixed for each model run for a given region. It must be in GeoTIFF format, and all grids must have same dimensions and bounding box. If no grid is specified, a uniform (constant-valued grid) is used. (Type: String, Default value: ./input_files/Horn_of_Africa_GPW-v4_pop_count_2020_450sec.tif)

Local connectivity grid file (C1_file): This is a spatial grid of values (2D array) that gives a measure of the "local connectivity", or potential for conflict to spread, in each grid cell. It may depend on indicators such as accessibility, road density, etc. It is fixed for each model run for a given region. It must be in GeoTIFF format, and all grids must have same dimensions and bounding box. If no grid is specified, a uniform (constant-valued grid) is used. (Type: String, Default value: blank space).

Nonlocal connectivity grid file (C2_file): This is a spatial grid of values (2D array) that gives a measure of the "nonlocal connectivity", or potential for conflict to spread to distant grid cells, in each grid cell. It depends on indicators such as internet and cell phone access. It is fixed for each model run for a given region. It must be in GeoTIFF format, and all grids must have same dimensions and bounding box. If no grid is specified, a uniform (constant-valued grid) is used. (Type: String, Default value: blank space).

Conflict emergence factor (c_emerge): This is a proportionality factor that is multiplied by values in the normalized unrest grid (U) to get the probability that conflict emerges in a given grid cell in the current timestep. (Type: Float, Min: 0.0, Max: 1.0, Default Value: 0.005).

Conflict local spreading factor (c_spread): This is a proportionality factor that is multiplied by values in the normalized local connectivity grid (C1) to get the probability that conflict spreads from a given grid cell to its nearest neighbors in the current timestep. (Type: Float, Min: 0.0, Max: 1.0, Default Value: 0.2).

Conflict nonlocal spreading factor (c_spread2): This is a proportionality factor that is multiplied by values in the normalized nonlocal connectivity grid (C2) to get the probability that conflict spreads from a given grid cell to remote grid cells in the current timestep. (Type: Float, Min: 0.0, Max: 1.0, Default Value: 0.0).

Conflict resolution probability (p_resolve): The probability that the conflict in any grid cell will be resolved in the current timestep. (Type: Float, Min: 0.0, Max: 1.0, Default Value: 0.47).

Start date (start_date): The start date for the simulation, in the standard form: YYYY-MM-DD. Time is measured from this date. (Type: String, Default Value: 2021-01-01.)

Time units (time_units): The amount of time that corresponds to one timestep. The parameters c_emerge, c_spread, c_spread2, and p_resolve are used to compute the probability that conflict will emerge, spread (locally or nonlocally) or be resolved in this amount of time. (Type: String [days, weeks, or months], Default value: days).

Explanation of Other Model Parameters

The Stochastic Conflict Model is set up with defaults that correspond to the Greater Horn of Africa, which has the geographic bounding box: (min_lat = -5, min_lon = 25.0, max_lat = 25.0,

max_lon = 55.0) and dimensions: (n_cols = 240, n_rows = 240) . This region spans 30 degrees of latitude and 30 degrees of longitude. However, the model can be applied to any geographic bounding box if the necessary input grids are available for that region.

The xsize of a model grid cell in arcseconds is given by 3600 * (max_lon - min_lon)/n_cols, where max_lon and min_lon have units of degrees. Similarly, the ysize of a model grid cell in arcseconds is given by 3600 * (max_lat - min_lat)/n_rows, where max_lat and min_lat have units of degrees. (Default Values: xsize = 450 arcseconds, ysize = 450 arcseconds. This corresponds to about 13.89 km near the equator.)

The following parameters can be changed in the model's configuration file, using a text editor. Here, the "long name" (in bold) is followed by the "key name" (in parentheses) that is used in the config file.

Number of columns (n_cols): The number of columns in the model grid. (Type: Integer, Min: 100, Max: 2000, Default Value: 240).

Number of rows (n_rows): The number of rows in the model grid. (Type: Integer, Min: 100, Max: 2000, Default Value: 240).

Minimum latitude (min_lat): The minimum latitude (south edge) of the model's geographic bounding box. The default bounding box values are for the Greater Horn of Africa. If GeoTIFF files are provided for U_file, C1_file, and/or C1_file, they must span this same bounding box.
(Type: Float, Min: -90.0, Max: 90.0, Default value: -5.0, Unit: degrees east).
Note: Use positive values north of the equator and negative values south of the equator.

Maximum latitude (max_lat): The maximum latitude (north edge) of the model's geographic bounding box. The default bounding box values are for the Greater Horn of Africa. If GeoTIFF files are provided for U_file, C1_file, and/or C1_file, they must span this same bounding box.
(Type: Float, Min: -90.0, Max: 90.0, Default Value: 25.0, Unit: degrees east.)
Note: Use positive values north of the equator and negative values south of the equator.

Minimum longitude (min_lon): The minimum longitude (west edge) of the model's geographic bounding box. The default bounding box values are for the Greater Horn of Africa. If GeoTIFF files are provided for U_file, C1_file, and/or C1_file, they must span this same bounding box.
(Type: Float, Min: -180.0, Max: 180.0, Default value: 25.0, Unit: degrees north).
Note: Use positive values east of the prime meridian and negative values west of it.

Maximum longitude (max_lon): The maximum longitude (east edge) of the model's geographic bounding box. The default bounding box values are for the Greater Horn of Africa. If GeoTIFF files are provided for U_file, C1_file, and/or C1_file, they must span this same bounding box.
(Type: Float, Min: -180.0, Max: 180.0, Default value: 55.0, Unit: degrees north).
Note: Use positive values east of the prime meridian and negative values west of it.

Explanation of Model Output Parameters

Note: The two primary outputs of the Stochastic Conflict Model are saved as a time-indexed set of geospatial grids spanning the modeled region, called a "grid stack". By default, these grid stacks are saved in the folder "~/outputs". Each model run generates one grid stack for the "conflict presence" variable and another for the "conflict_IDs" variable (used for better visualizations). In addition, each grid stack is saved in two different file formats:

(1) RTS (RiverTools Sequence) format, row-major order, IEEE binary, 4-byte floats, with georeferencing in an RTI (RiverTools Information) header file, and

(2) NetCDF format, with standardized metadata, closely following the CF (Climate and Forecasting) Conventions.

The RTS format is a simple, efficient, binary format that is easy to read into a variety of applications.

The following parameters can be changed in the model's configuration file, using a text editor. Here, the "long name" (in bold) is followed by the "key name" (in parentheses) that is used in the config file.

Option to allow overwriting existing output files (OVERWRITE_OK): Set this to 1 to allow existing output files with the same filename to be overwritten, and 0 otherwise. If set to zero, numbers will be inserting in the existing filename before the extension. For example, conflict.rts, conflict_1.rts, conflict_2.rts. Note that all output files will be converted to movies after each model run, so it is usually better to set this to 1. (Type: Integer, Min: 0, Max: 1, Default Value: 1).

Conflict presence output file (out_file): The filename (with path) of an output file that stores a grid stack of time-indexed grids that contain zeros and ones. A grid cell has the value 1 if it is experiencing conflict at that time and has the value 0 otherwise. This filename must end in ".rts", but output will also be stored in NetCDF format with the same filename but extension "_2D.nc". (Type: String, Default Value: ~/output/conflicts.rts).

Conflict IDs output file (IDs_file): The filename (with path) of an output file that stores a grid stack of time-indexed grids that contain long-integer IDs. All grid cells that obtained conflict by spreading from the same initial conflict are assigned the same unique ID number to allow visualization. This filename must end in ".rts", but output will also be stored in NetCDF format with the same filename but extension "_2D.nc". An ID of 0 is assigned to grid cells with no conflict, and an ID of 10000000 is assigned to grid cells for which there is no unrest (i.e., no potential for conflict), such as lake and ocean grid cells. (Type: String, Default Value: ~/output/conflict_IDs.rts).

Option to create MP4 movies (CREATE_MP4_MOVIES): Set this to 1 to create MP4-format movies from model output files at the end of the simulation, or 0 to not create them. (Type: Integer, Min: 0, Max: 1, Default Value: 1).

Option to create WEBM movies (CREATE_WEBM_MOVIES): Set this to 1 to create WEBM-format movies from model output files at the end of the simulation, or 0 to not create them. (Type: Integer, Min: 0, Max: 1, Default Value: 1). The WEBM format supports opacity.

Opacity of media files (opacity): This determines the opacity to apply to the visualization files so that they will not obscure city names and boundaries in a Causemos basemap. A value of 1.0 corresponds to completely opaque (not transparent at all). (Type: Float, Min: 0.0, Max: 1.0, Default Value: 0.7).

Conflict presence movie w/o opacity: This filename is auto-generated from the "Conflict presence output file" filename (see above) by changing the filename extension of that NetCDF file to ".mp4". The grid stack in the NetCDF file is converted to a set of color images that are saved as a movie in MP4 format. Movie files are saved in the "~/media/movies" folder.

Conflict presence movie w/ opacity: This filename is auto-generated from the "Conflict presence output file" filename (see above) by changing the filename extension of that NetCDF file to ".webm". The grid stack in the NetCDF file is converted to a set of color images with the specified opacity (see above) that are saved as a movie in WEBM format. Note that the WEBM movie format supports opacity while the MP4 format does not. Opacity allows the movie to be displayed over a backdrop map that shows country borders and cities, as in Causemos, to simplify interpretation. Movie files are saved in the "~/media/movies" folder.

Conflict IDs movie w/o opacity: This filename is auto-generated from the "Conflict IDs output file" filename (see above) by changing the filename extension of that NetCDF file to ".mp4". The grid stack in the NetCDF file is converted to a set of color images that are saved as a movie in MP4 format. Movie files are saved in the "~/media/movies" folder.

Conflict IDs movie w/ opacity: This filename is auto-generated from the "Conflict IDs output file" filename (see above) by changing the filename extension of that NetCDF file to ".webm". The grid stack in the NetCDF file is converted to a set of color images with the specified opacity (see above) that are saved as a movie in WEBM format. Note that the WEBM movie format supports opacity while the MP4 format does not. Opacity allows the movie to be displayed over a backdrop map that shows country borders and cities, as in Causemos, to simplify interpretation. Movie files are saved in the "~/media/movies" folder.

Different Ways to Run the Model

The Stochastic Conflict Model is distributed as an open-source Python package in a GitHub repository at:

https://github.com/peckhams/stochastic_conflict_model/

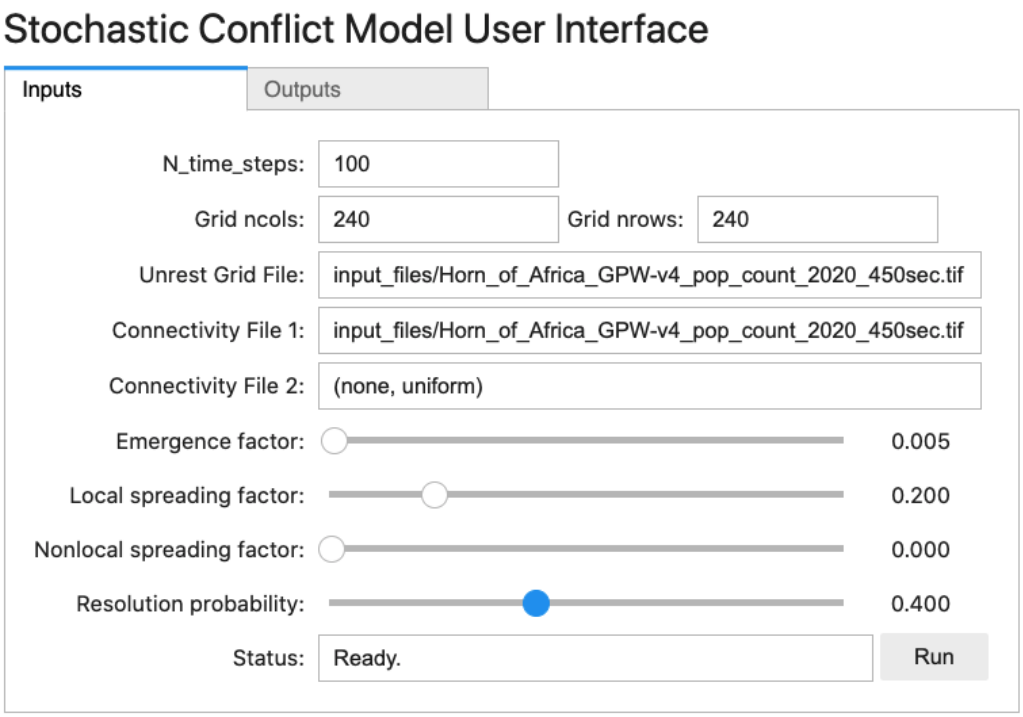
Instructions for installing the Python package are given in **Appendix 1** of a Jupyter notebook that is included with the package in the *notebooks* folder. The URL for that notebook is:

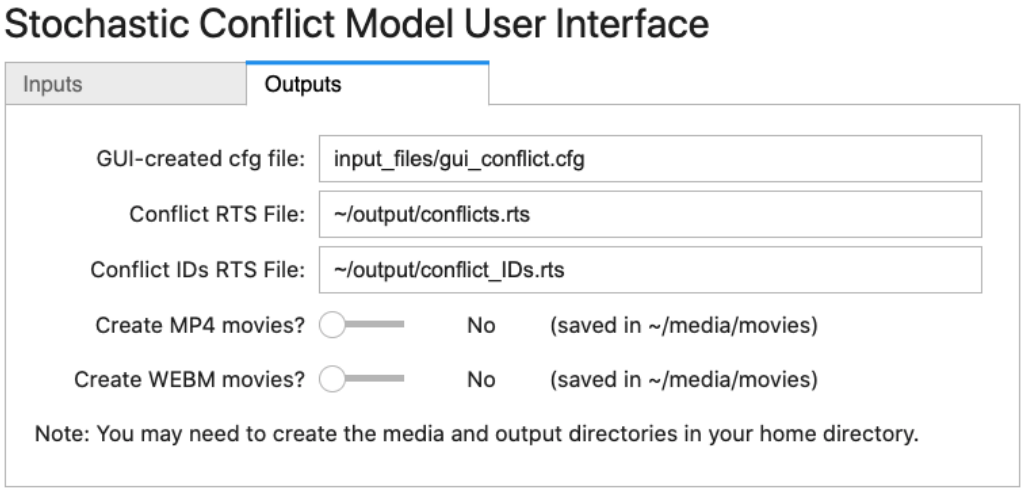
https://github.com/peckhams/stochastic_conflict_model/blob/main/notebooks/Conflict_Introduction.ipynb

The model reads its parameters from a configuration file that has the filename extension ".cfg". This is a simple text file with key-value pairs that can be edited with any plain text editor. Three examples are given in a folder called "input_files" that is included in the Python package.

Jupyter Notebook GUI

The Jupyter notebook includes a graphical user interface (GUI), with sliders and text boxes, for running the model. Values entered or selected in the GUI are written to the model's configuration file. You then click the Run button to run the model. Model output can also be visualized within the notebook with special graphics routines that are included with the package. The next two images show the input and output panels of the tabbed GUI.





Terminal Prompt (Mac or Linux)

Assuming the Python package has been installed, the model can be run from a terminal prompt with the following commands. If you have installed the package in a conda environment called "conflict", first switch to that environment with: "conda activate conflict", then type:

```
% cd stochastic_conflict_model
% python conflict --cfg_file './input_files/conflict.cfg'
```

Python Prompt

Assuming the Python package has been installed, the model can be run from a Python prompt. First, open a terminal window. If you have installed the package in a conda environment called "conflict", next switch to that environment by typing: "conda activate conflict". Finally, start a Python command-line session by typing "python", and then type the following commands at the Python prompt:

```
>>> import os
>>> src_dir = '~ /stochastic_conflict_model'
>>> os.chdir( src_dir )

>>> from conflict.model import conflict
>>> c = conflict.conflict()

>>> cfg_file = 'input_files/conflict.cfg'
>>> c.run_model( cfg_file=cfg_file  )
```

As a container with Dojo/Causemos

The Stochastic Conflict Model has been published in Causemos, from Uncharted. Causemos aggregates model predictions to administrative zones, so results for countries not fully contained in the model's geographic bounding box may be misleading. For details, go to: https://causemos.uncharted.software. Note that a login is required. See Appendix 2 for details on how the model was containerized and then annotated with Dojo from Jataware.

Examples for the Greater Horn of Africa

In this section we illustrate the range of behavior that the model can exhibit with a series of specific examples for the ***Greater Horn of Africa region***. For each of these examples, a ***GPW-v4 population count*** grid is used for both the unrest grid, U, and connectivity grid, C1. See Figure 1.

Note that the GPW-v4 data set has large nodata regions for the country of Egypt, which affects the top left part of the model's default domain. This issue is explained at: https://sedac.uservoice.com/knowledgebase/articles/799203-why-does-egypt-look-different-than-the-rest-of-nor.

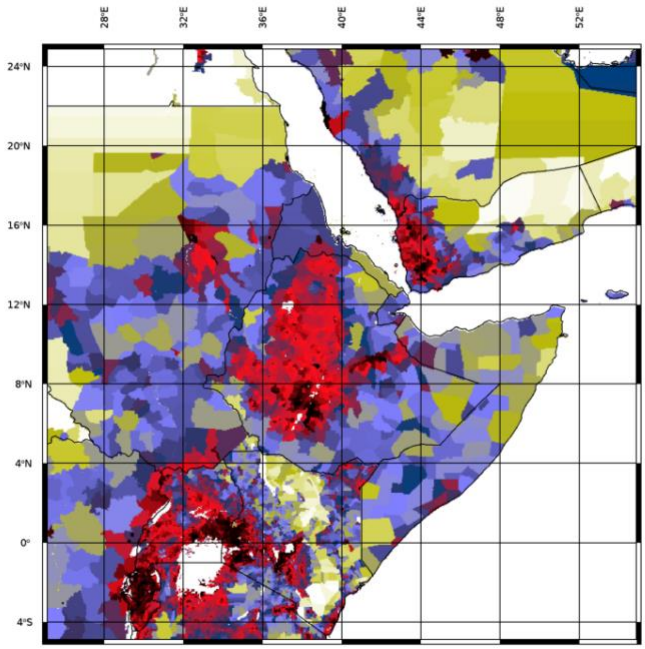


Figure 1. Color image for the GPW-v4 population count grid with a "Stern special" color table. Here, white and yellow indicate low values and red and black indicate high values. Notice the high populations in Ethiopia and Sudan (middle), around Lake Victoria (bottom left: Uganda, Kenya, Tanzania, Burundi, and Rwanda) and the southwest part of the Arabian Peninsula (Saudi Arabia and Yemen). The ocean, lakes and nodata regions (top) are all white. Polygons show true resolution of source data, typically much coarser than the 30-arcsec resolution of the data set.

Examples with No Spreading

Case 1: U = Uniform, No Spreading. The case of uniform U (spatially uniform and unchanging in time) with no spreading is mathematically tractable (see Appendix 3). The main result in this case is that the probability of conflict in any grid cell, or P[S(k) = 1], converges rapidly to:

$$f = pe / (pe + pr) \quad (1)$$

where pe = p_emerge and pr = p_resolve. For this case, p_emerge = c_emerge. It follows that the expected fraction of grid cells with conflict should also approach the same number, f. Solving (1) for pe, gives: pe = pr (f / (1-f)). This shows that for any given value of pr < 1, there is a corresponding value of pe that results in the same value of the fraction, f. If we change variables to f = 1/n, then we have: pe = pr / (n-1). If n=10, we have f=1/10 and pe = pr / 9. There are

infinitely many (pr, pe) pairs that satisfy this constraint, two of which are (pr, pe) = (0.9, 0.1) and (pr, pe) = (0.09, 0.01). Figures 2a and 2b show these two special cases, which do appear to be visually and statistically similar. However, the conflicts in the second case (Figure 2b, pr=0.09, pe=0.01) are 10 times less likely to emerge than those in the first case (Figure 2a, pr=0.9, pe=0.1), but last for 10 times longer, on average.

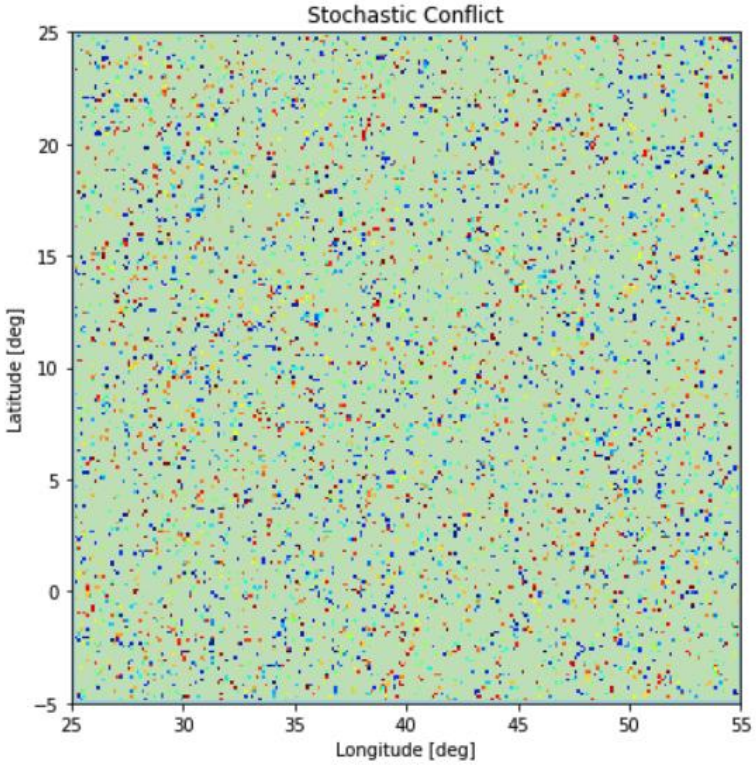


Figure 2a. Uniform, no spreading, ce=0.1, cs1=0, cs2=0, pr=0.9, n=100).

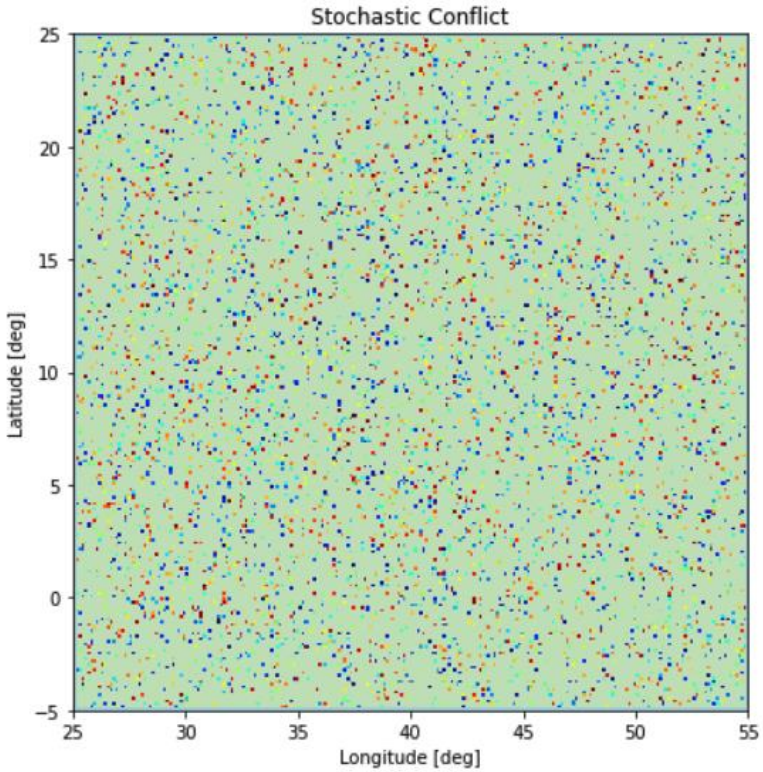


Figure 2b. Uniform, no spreading, ce=0.01, cs1=0, cs2=0, pr=0.09, n=100).

Case 2: U = Population Count, No Spreading

We now continue with the case of no spreading, but with the U grid set equal to the population count (GPW v4).

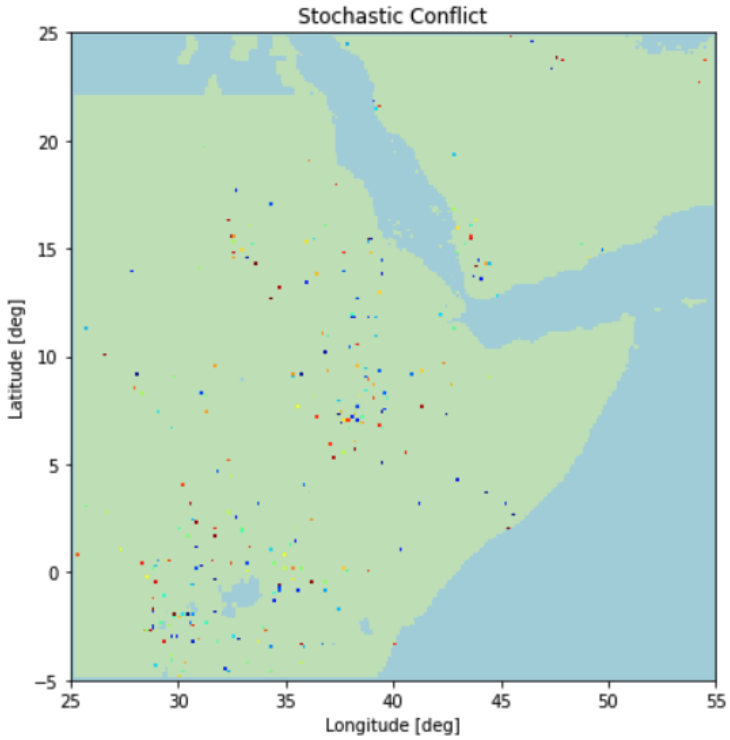


Figure 3a. No spreading, ce=0.1, cs1=0, cs2=0, pr=0.1, n=100).

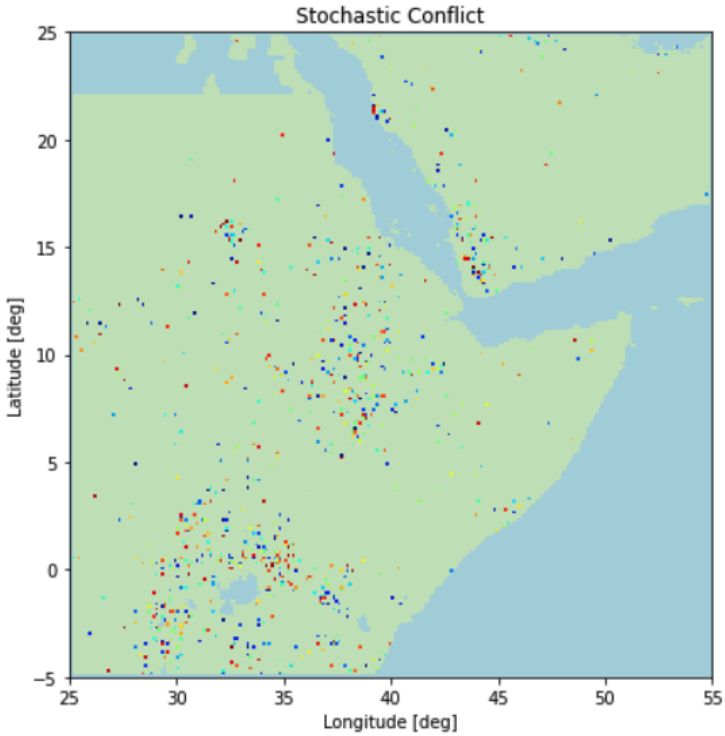


Figure 3b. No spreading, ce=0.4, cs1=0, cs2=0, pr=0.1, n=100).

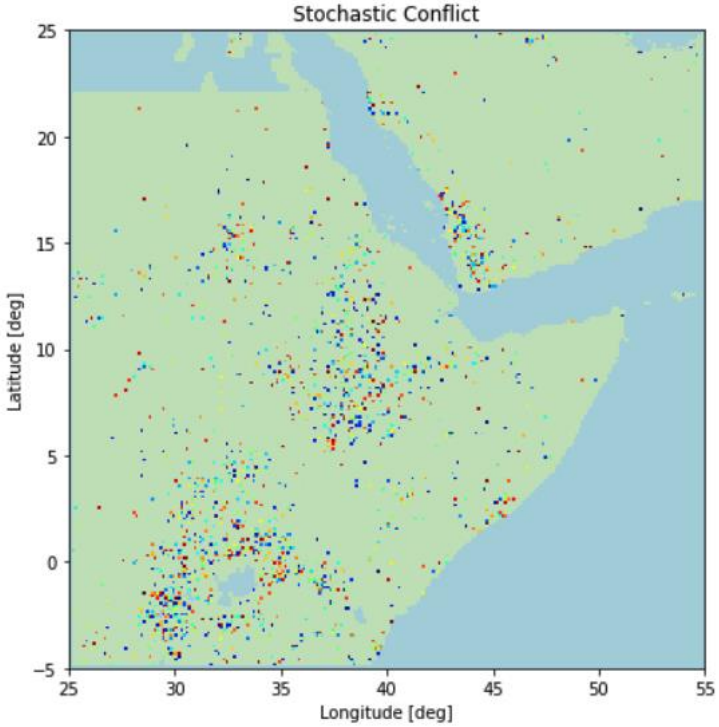


Figure 3c. No spreading, ce=0.9, cs1=0, cs2=0, pr=0.1, n=100).

Examples with Local Spreading Only

Case 3: U = C1 = Uniform. Again, we start with the special case where U and C1 are uniform grids. We use the same parameters as in Figures 2a and 2b, but we increase the local spreading factor, first to the same small value of 0.03, and then to 0.3 for one of them so that all 3 parameters are scaled by a factor of 10. Despite the similarity of the two cases in the absence of spreading, now they all look quite different.

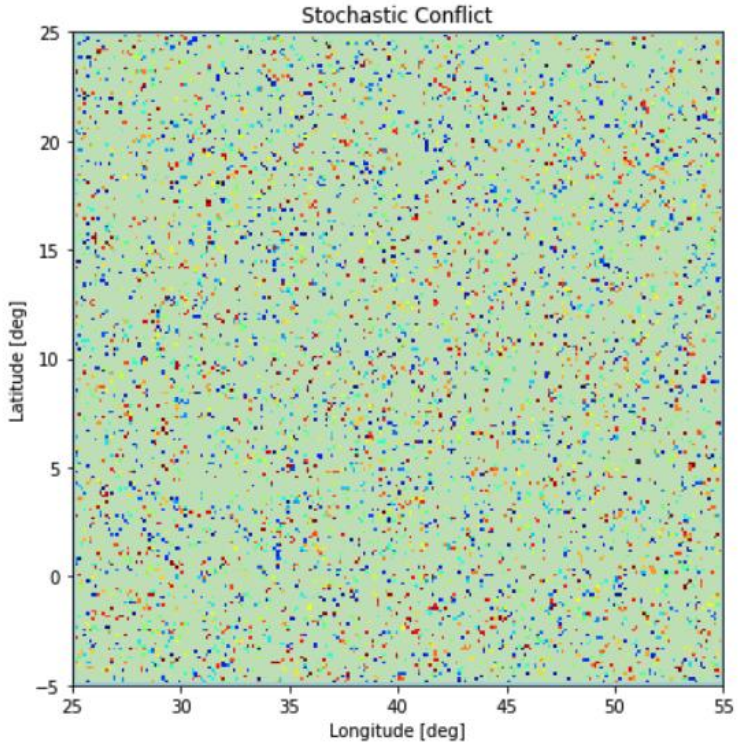


Figure 4a. Uniform U and C1, with local spreading (ce=0.1, cs1=0.03, cs2=0, pr=0.9, n=100).

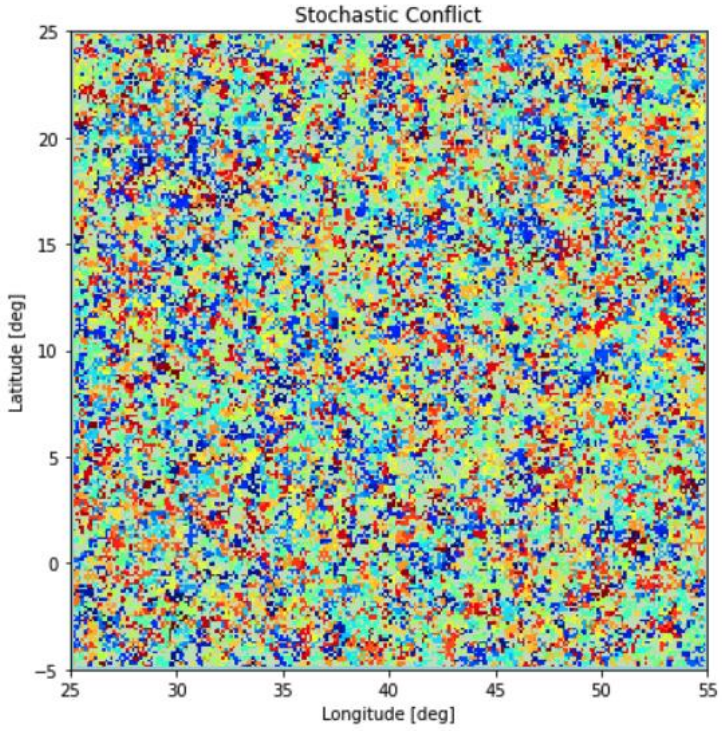


Figure 4b. Uniform U and C1, with local spreading (ce=0.01, cs1=0.03, cs2=0, pr=0.09, n=100).

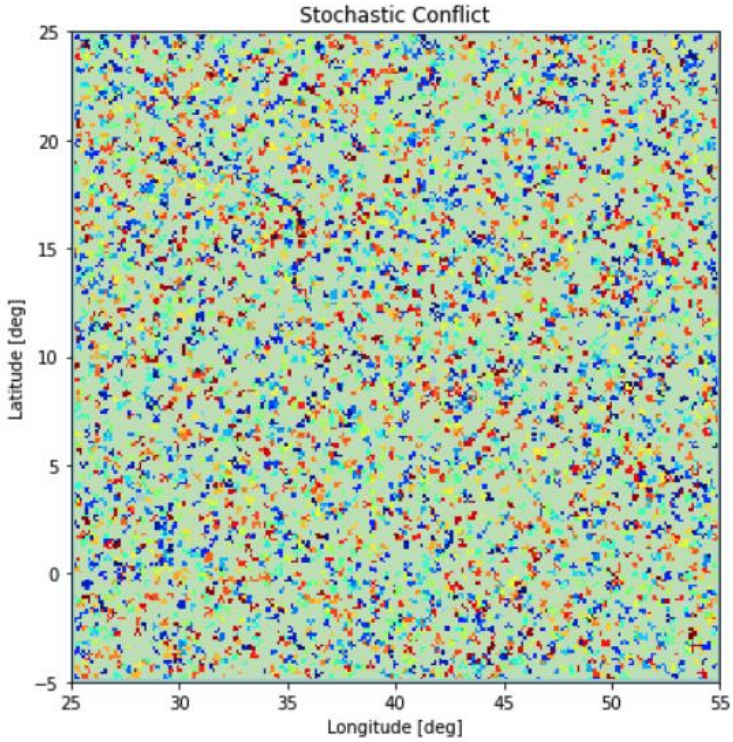


Figure 4c. Uniform U and C1, with local spreading (ce=0.1, cs1=0.3, cs2=0, pr=0.9, n=100).

Case 4: U = C1 = Population Count, ce = 0.4, cs1 = 0.04, pr Variable

Figures 5a, 5b, 5c, and 5d all have a fairly high probability of conflict emergence (ce = 0.4), a small amount of local spreading (cs1 = 0.04), no nonlocal spreading (cs2 = 0), and conflict resolution probabilities of 0.1, 0.2, 0.3 and 0.4. The average size and total number of conflict clusters both decrease as the probability of conflict resolution increases.

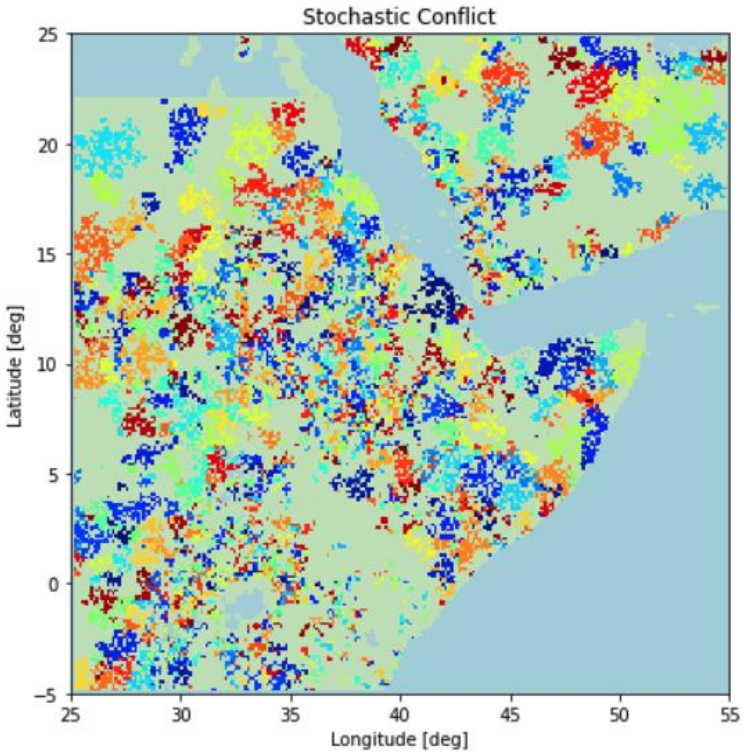


Figure 5a. Local spreading, with ce=0.4, cs1=0.04, cs2=0, pr=0.1, n=100.

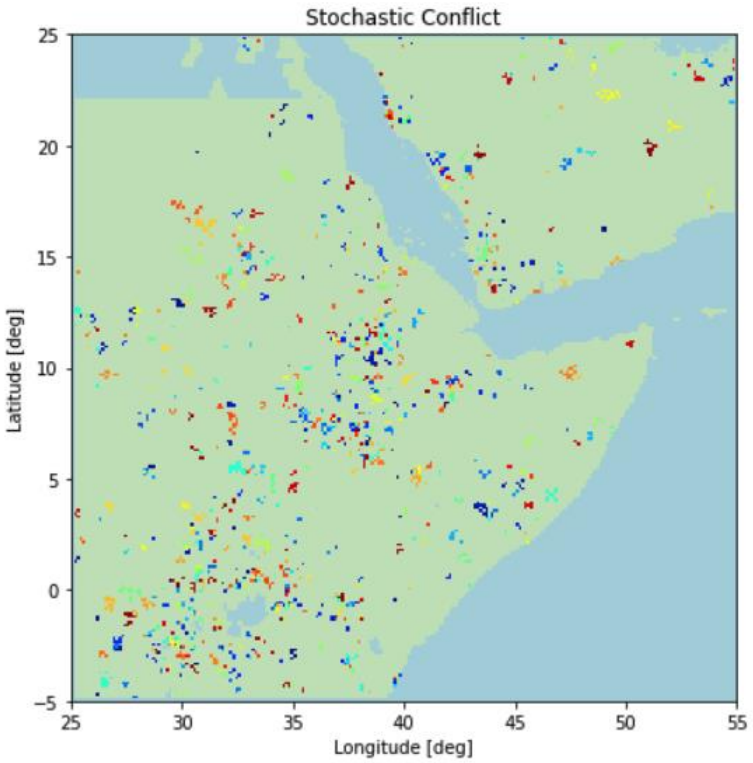


Figure 5b. Local spreading, with ce = 0.4, cs1 = 0.04, cs2 = 0, pr = 0.2, n=100.

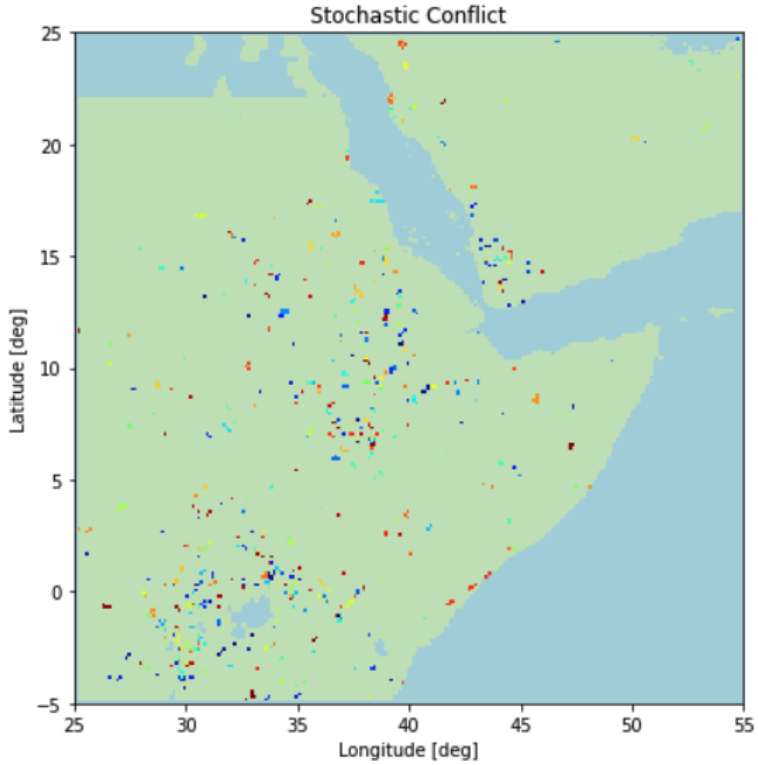


Figure 5c. Local spreading, with ce = 0.4, cs1 = 0.04, cs2 = 0, pr = 0.3, n=100.

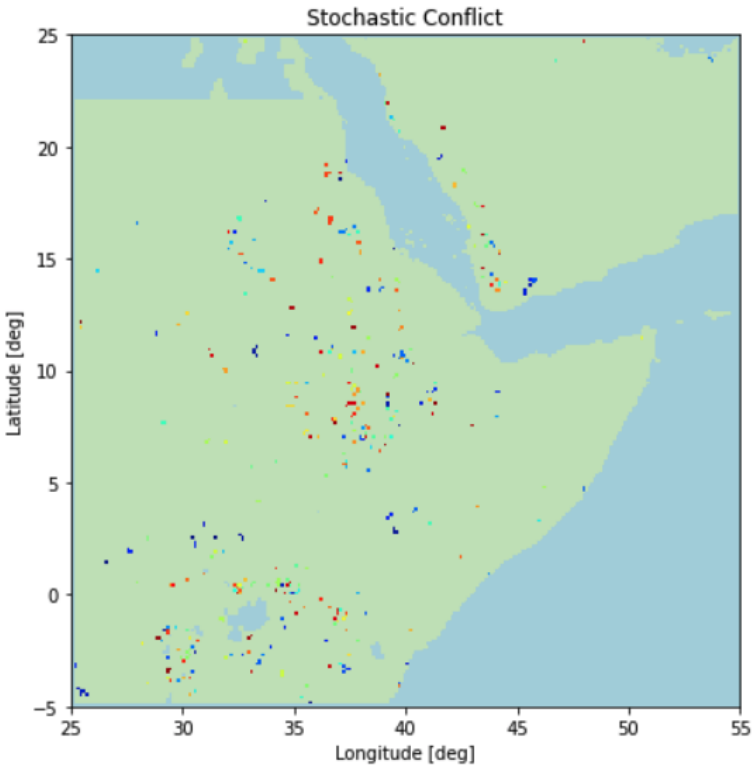


Figure 5d. Local spreading, with ce = 0.4, cs1 = 0.04, cs2 = 0, pr = 0.4, n=100.

Case 5: U = C1 = Population Count, ce = 0.03, cs1 = 0.2, Variable pr

Figures 6a, 6b, 6c, 6d, and 6e all have a small probability of conflict emergence (ce = 0.05), a significant probability of local spreading (cs1 = 0.3), no nonlocal spreading (cs2 = 0) and resolution probabilities, pr, of 0.2, 0.3, 0.4, 0.5, and 0.6.

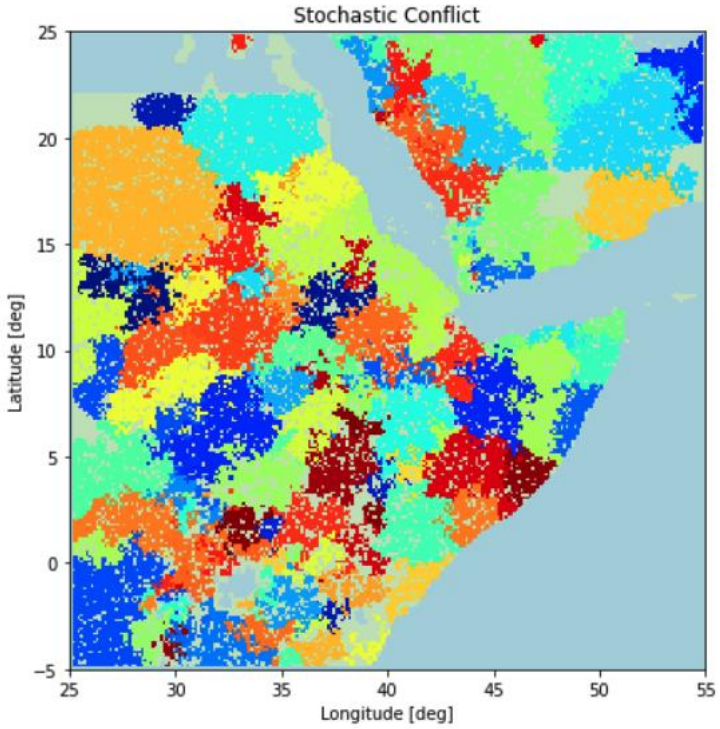


Figure 6a. Local spreading, with ce = 0.03, cs1 = 0.2, cs2 = 0, pr = 0.2, n=100.

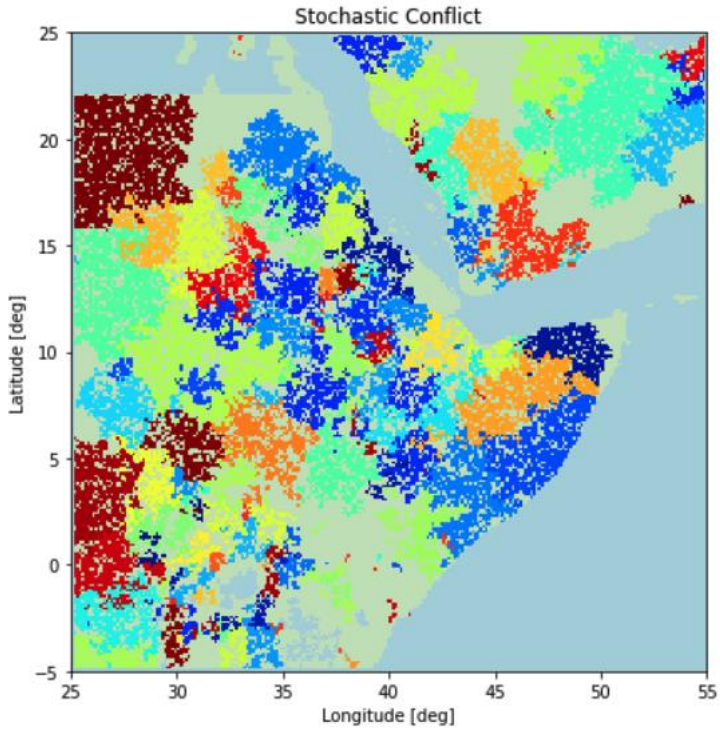


Figure 6b. Local spreading, with ce = 0.03, cs1 = 0.2, cs2 = 0, pr = 0.3, n=100.

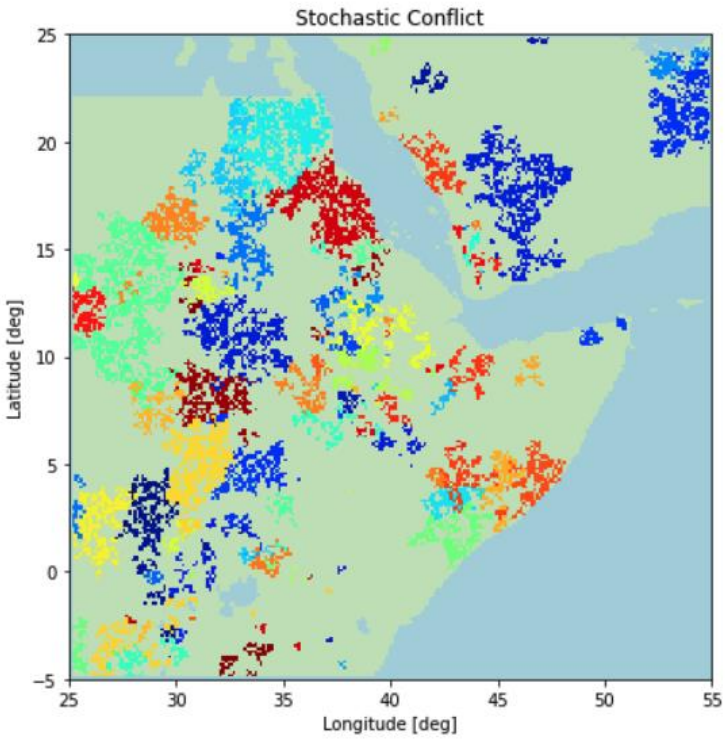


Figure 6c. Local spreading, with ce = 0.03, cs1 = 0.2, cs2 = 0, pr = 0.4, n=100.

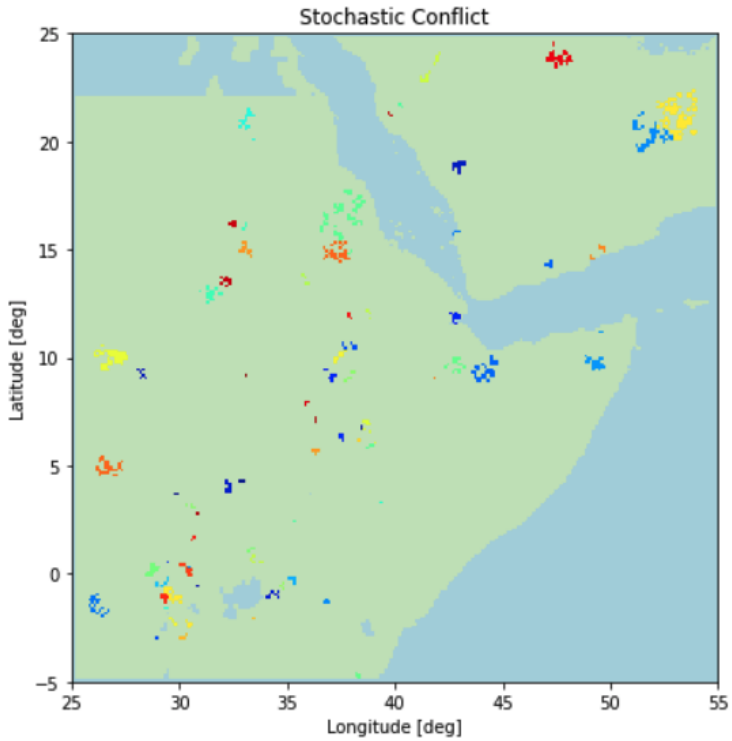


Figure 6d. Local spreading, with ce = 0.03, cs1 = 0.2, cs2 = 0, pr = 0.5, n=100.

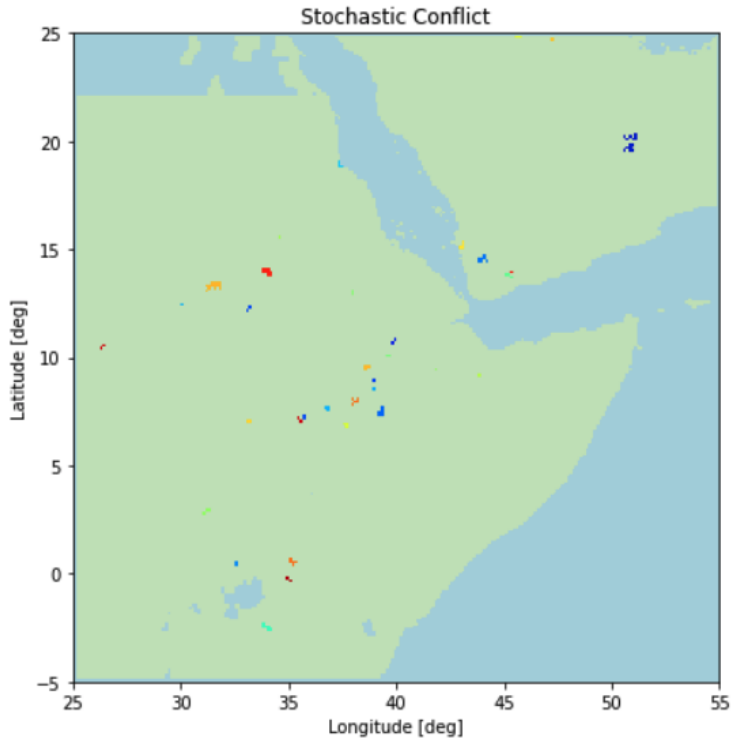


Figure 6e. Local spreading, with ce = 0.03, cs1 = 0.2, cs2 = 0, pr = 0.6, n=100.

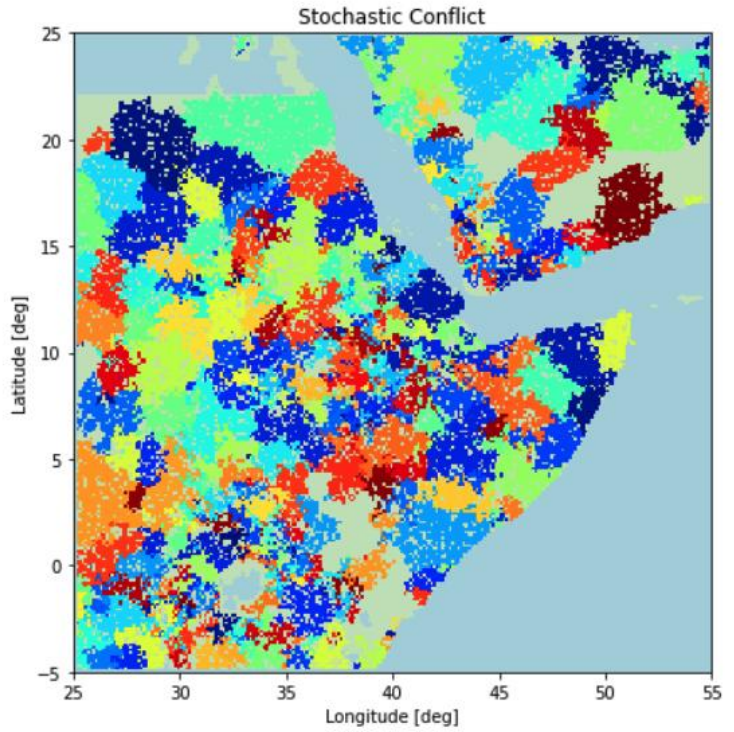


Figure 6f. Local spreading, with ce = 0.1, cs1 = 0.1, cs2 = 0, pr = 0.1, n=100.

Examples with Nonlocal Spreading Only

Case 6: U = C1 = Population Count, ce = 0.001, cs1 = 0, cs2=1, Variable pr

Figures 7a, 7b, … show cases with a very small probability of conflict emergence (ce = 0.001), no local spreading (cs1 = 0), and certain nonlocal spreading (cs2 = 1), with various resolution probabilities.

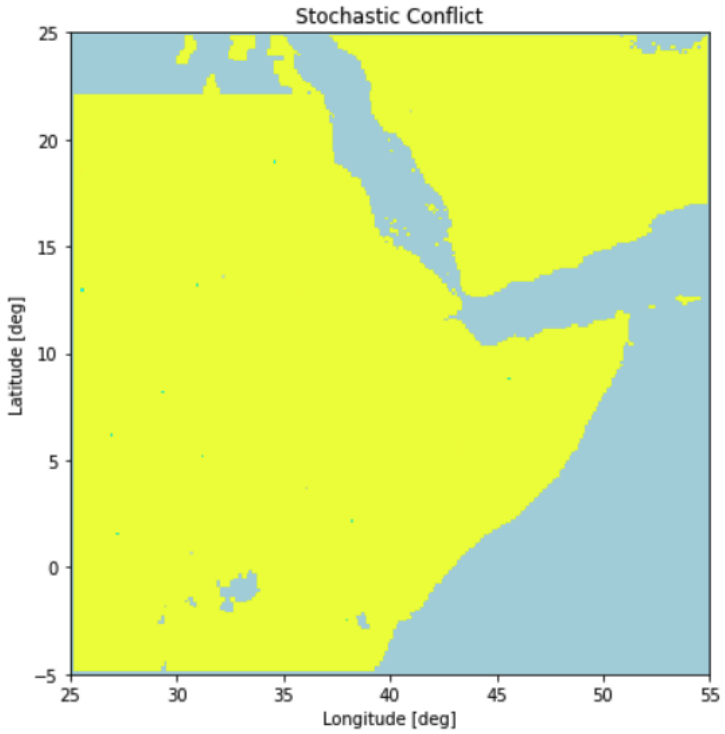


Figure 7a. Nonlocal spreading only, with ce=0.001, cs1=0, cs2=1, pr=0.

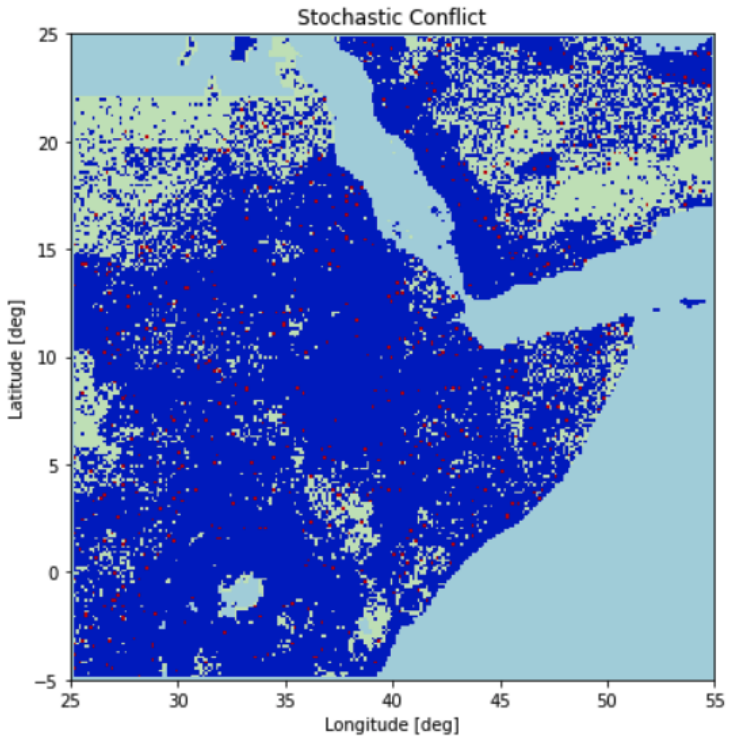


Figure 7b. Nonlocal spreading only, with ce=0.001, cs1=0, cs2=1, pr=0.001.

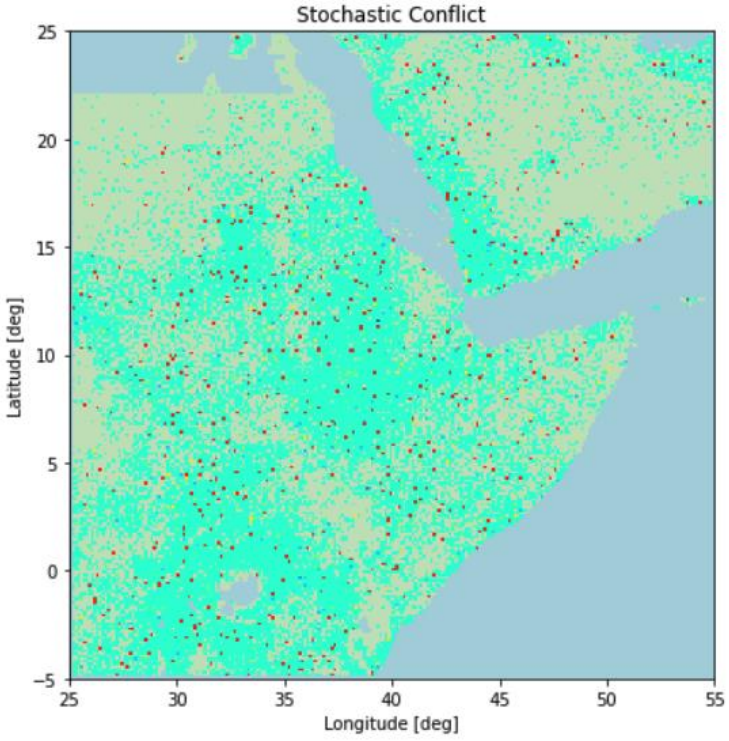


Figure 7c. Nonlocal spreading only, with ce=0.001, cs1=0, cs2=1, pr=0.01.

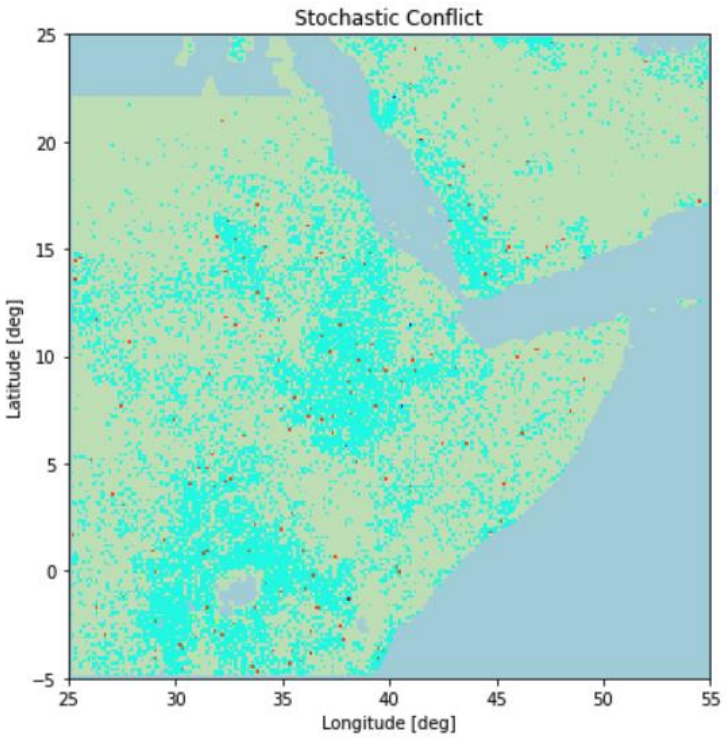


Figure 7d. Nonlocal spreading only, with ce=0.001, cs1=0, cs2=1, pr=0.03.

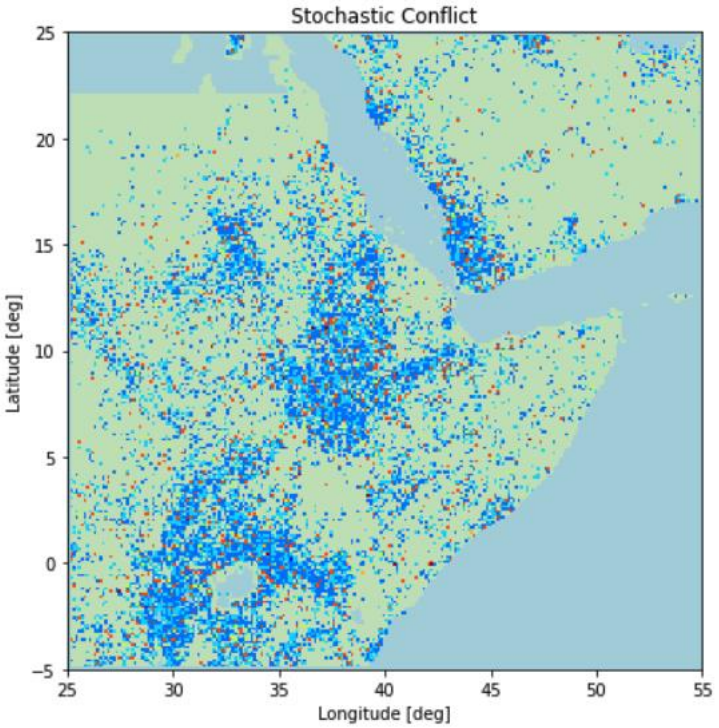


Figure 7e. Nonlocal spreading only, with ce=0.001, cs1=0, cs2=1, pr=0.05.

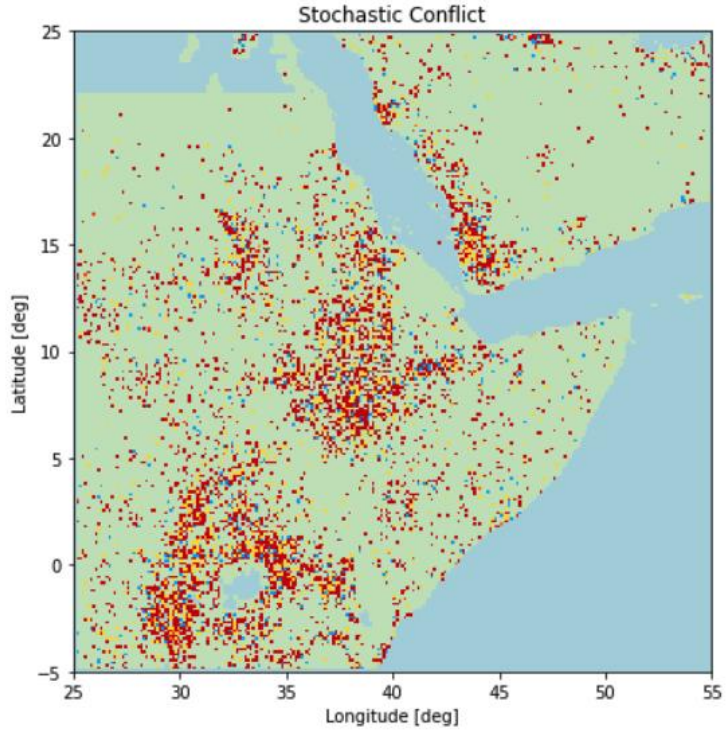


Figure 7f. Nonlocal spreading only, with ce=0.001, cs1=0, cs2=1, pr=0.1.

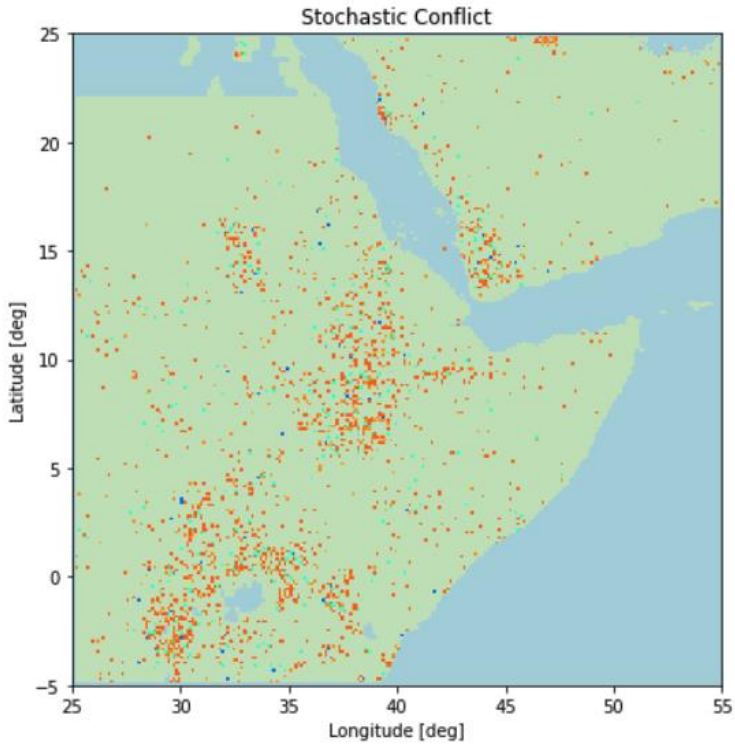


Figure 7g. Nonlocal spreading only, with ce=0.001, cs1=0, cs2=1, pr=0.3.

Case 6b: U = C1 = Population Count, Variable ce, cs1 = 0, cs2=0.5, pr=0.1

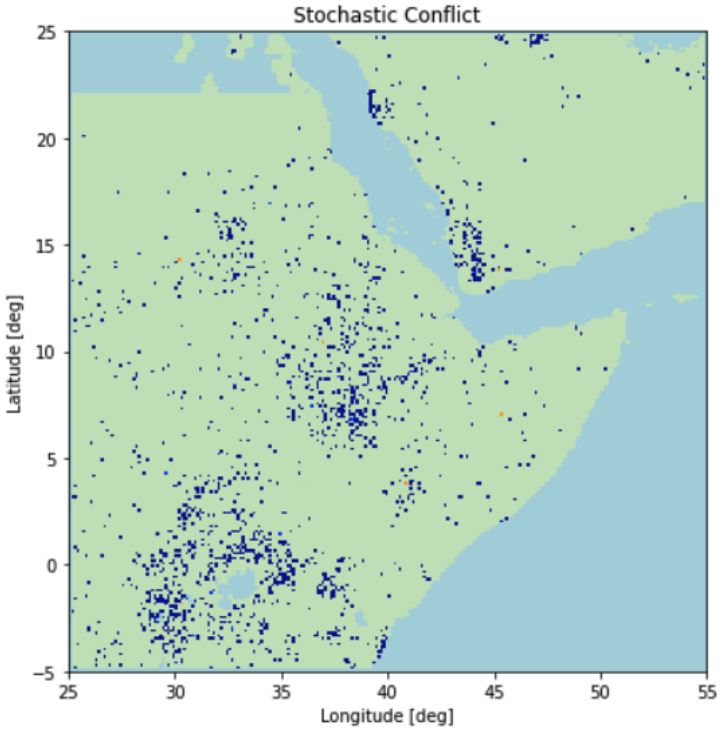


Figure 7h. Nonlocal spreading only, with ce=0.001, cs1=0, cs2=0.5, pr=0.1.

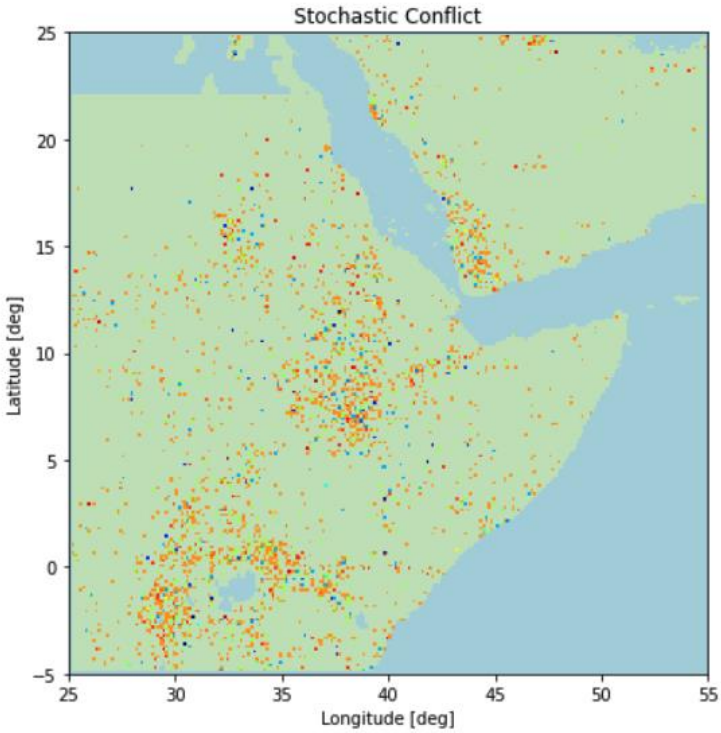


Figure 7i. Nonlocal spreading only, with ce=0.01, cs1=0, cs2=0.5, pr=0.1.

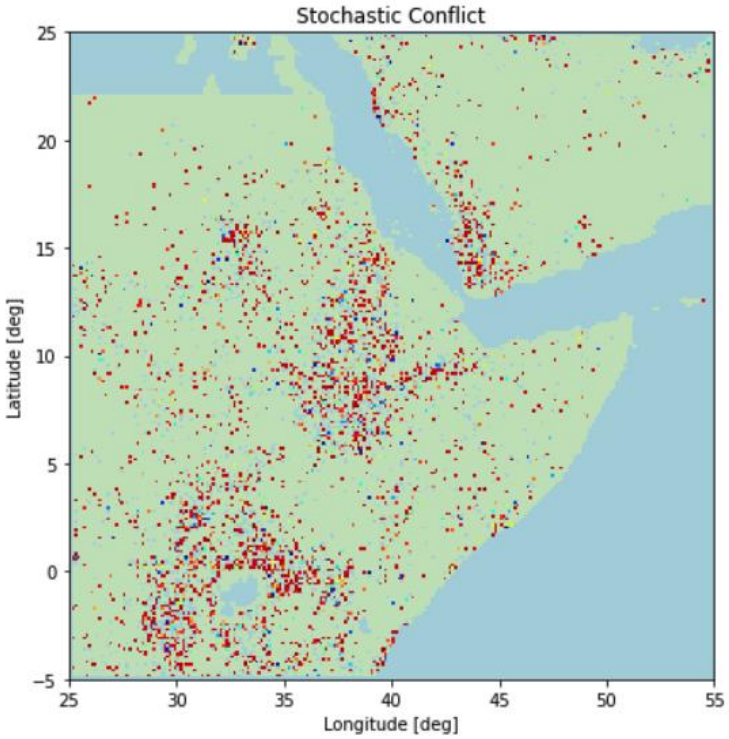


Figure 7j. Nonlocal spreading only, with ce=0.1, cs1=0, cs2=0.5, pr=0.1.

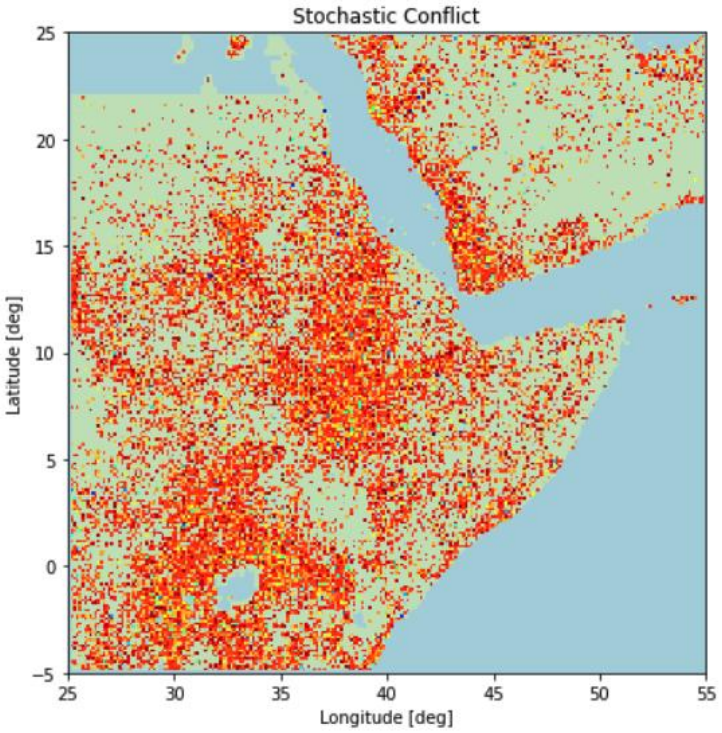


Figure 7k. Nonlocal spreading only, with ce=0.5, cs1=0, cs2=0.5, pr=0.1.

Examples with Local and Nonlocal Spreading

Case 7: U = C1 = Population Count, ce = 0.2, cs1 = 0.1, cs2=0.1, Variable pr

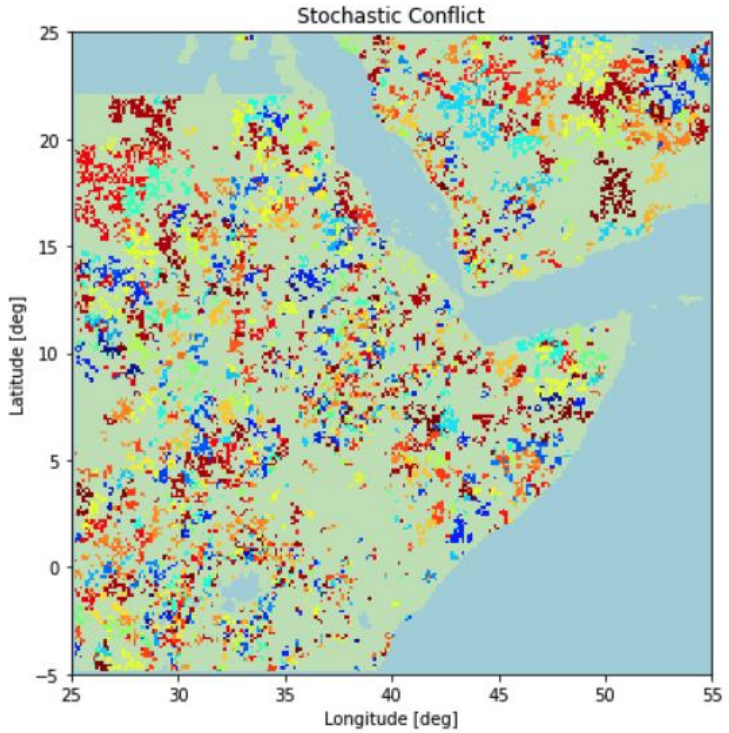


Figure 8a. Local & nonlocal spreading (ce=0.2, cs1=0.1, cs2=0.1, pr=0.3).

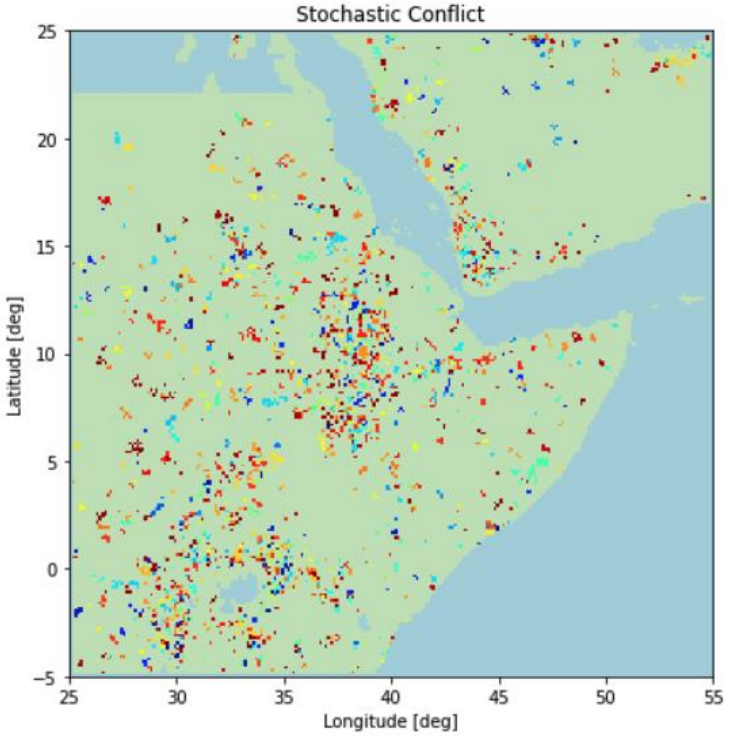


Figure 8b. Local & nonlocal spreading (ce=0.2, cs1=0.1, cs2=0.1, pr=0.4).

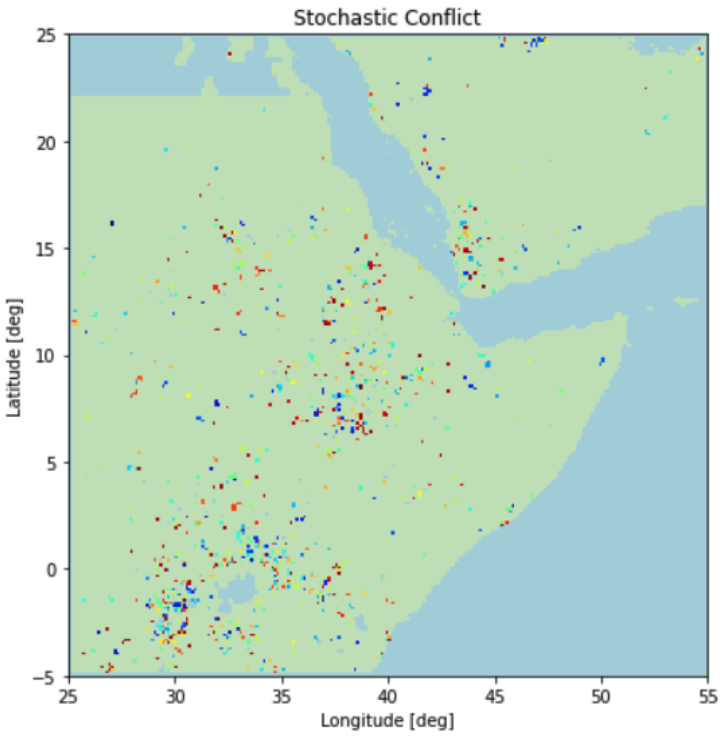


Figure 8c. Local & nonlocal spreading (ce=0.2, cs1=0.1, cs2=0.1, pr=0.6).

Case 8: U = C1 = Population Count, ce = 0.2, Mixed Local & Nonlocal Spreading

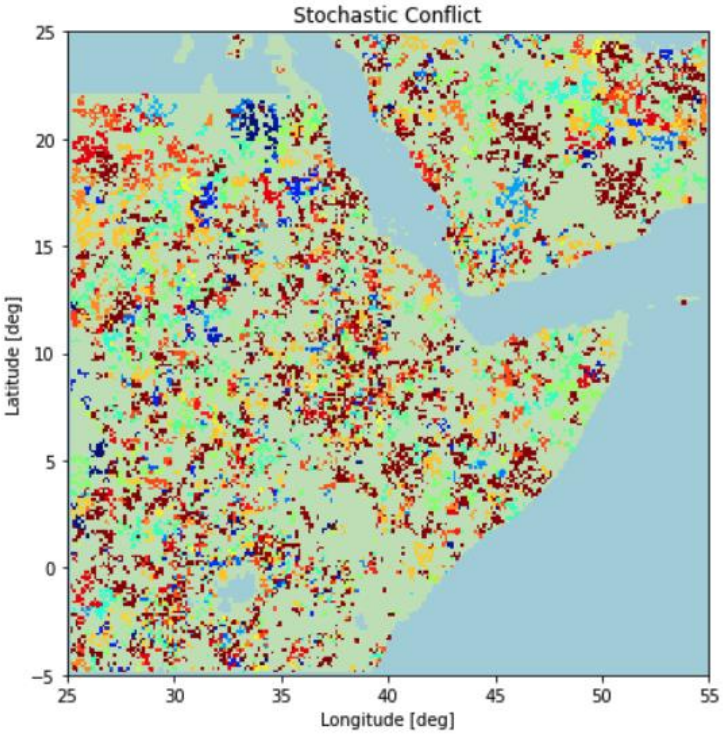


Figure 9a. Local & nonlocal spreading (ce=0.2, cs1=0.1, cs2=0.2, pr=0.3).

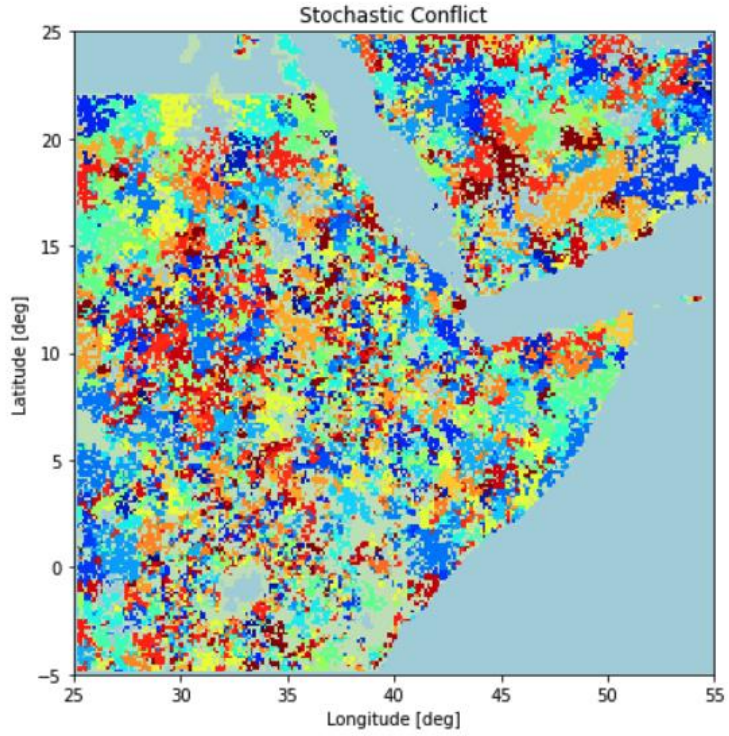


Figure 9b. Local & nonlocal spreading (ce=0.2, cs1=0.2, cs2=0.1, pr=0.3).

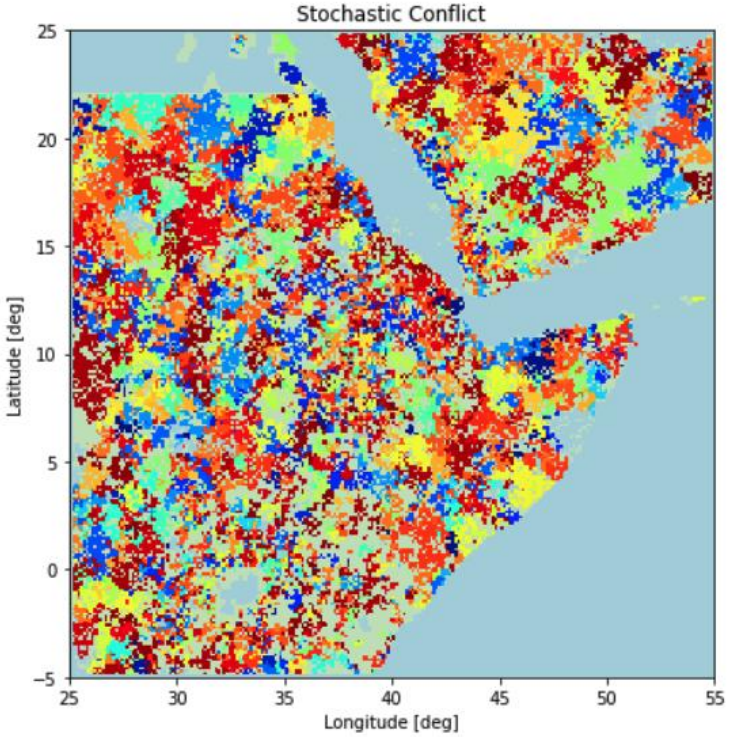


Figure 9c. Local & nonlocal spreading (ce=0.5, cs1=0.2, cs2=0.1, pr=0.3).

Case 9: U = Inverse Distance to Border (ce = 0.01, cs2=0, pr=0.6)

The figures in this section were generated by using a grid of inverse distance to own border as the Unrest grid file and a small conflict emergence factor. See the section: Preparing Input Grids for the Model for details.

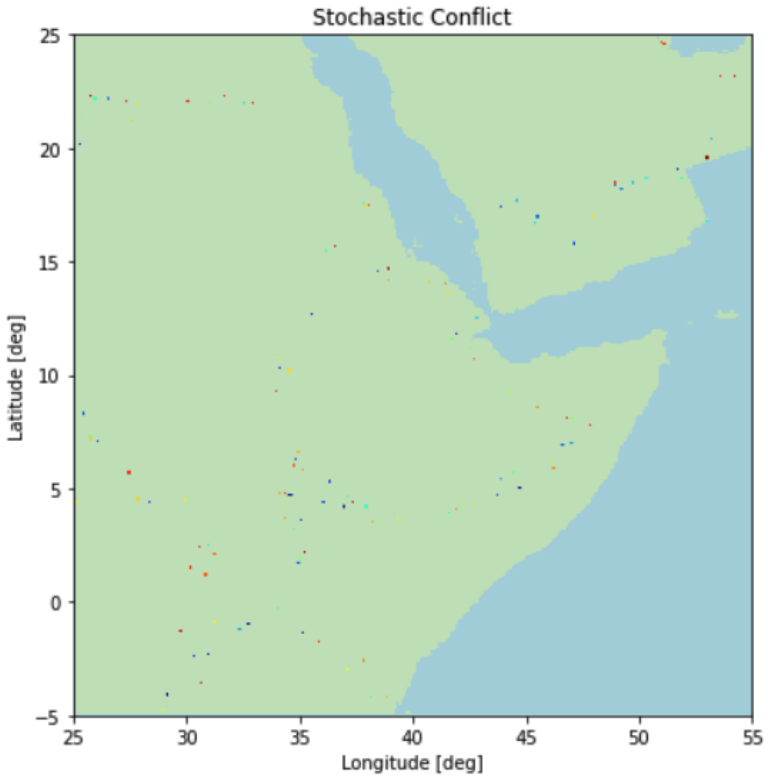


Figure 10a. Inverse distance (ce=0.0.1, cs1=0, cs2=0, pr=0.6).

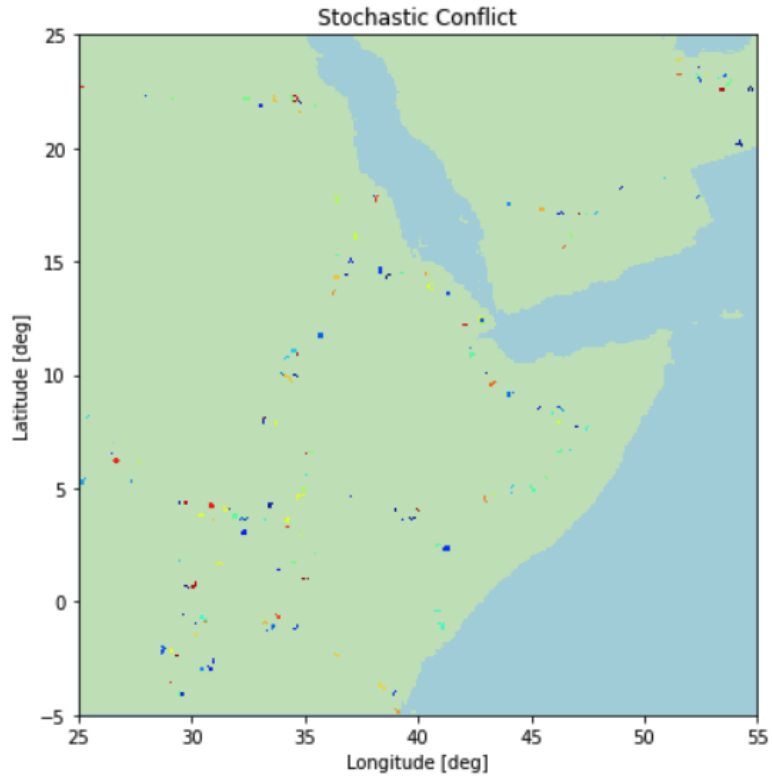


Figure 10b. Inverse distance (ce=0.01, cs1=0.1, cs2=0, pr=0.6). Small local spreading.

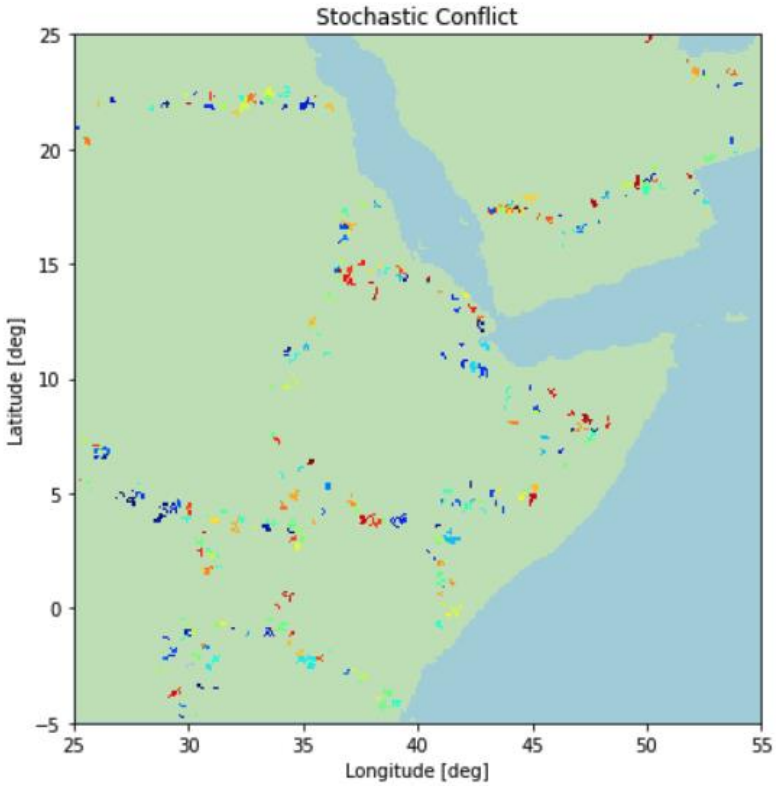


Figure 10c. Inverse distance (ce=0.01, cs1=0.2, cs2=0, pr=0.6). Increased local spreading.

Case 10: U = Inverse Distance to Border (ce = 0.2, cs2=0, pr=0.6)

This case shows a much larger conflict emergence factor and a high probability of conflict resolution. While unrealistic, it illustrates the high likelihood for conflict to emerge along borders between countries.

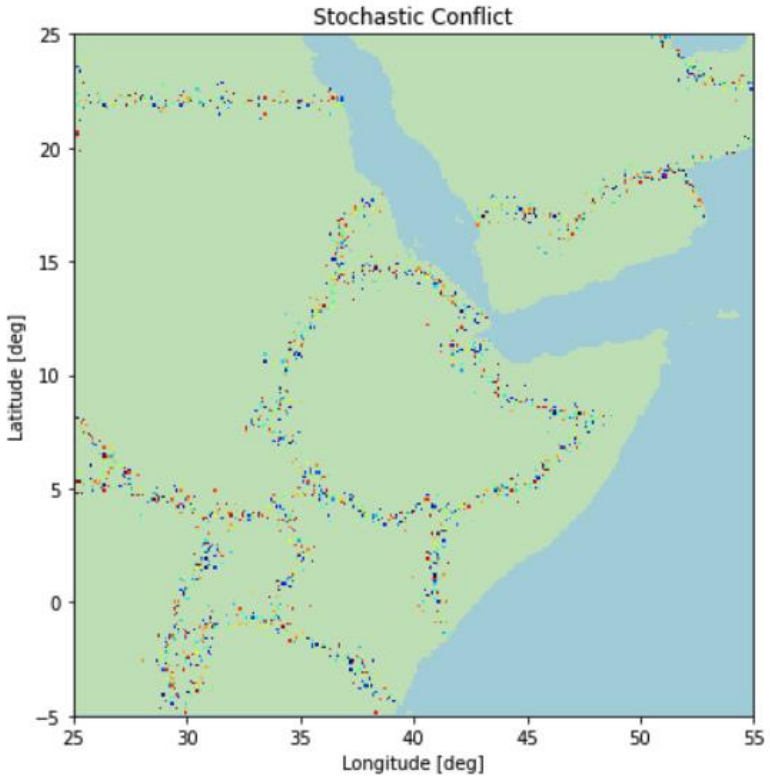


Figure 11a. Inverse distance (ce=0.2, cs1=0.0, cs2=0, pr=0.6).

Comparison to ACLED Data

ACLED (Armed Conflict Location and Event Data, acleddata.com) is one of the best-available sources of data for global conflicts, and in could, in principle, be used to calibrate the Stochastic Conflict Model (SCM). The ACLED data set is distributed as a very large CSV file or Excel spreadsheet. The SCM Python package contains a folder called "conflict/utils" that includes a collection of utilities that were written to perform various lower-level tasks. Among these there is a file called "acled.py" that provides Python functions for working with ACLED data. It includes a function called "save_acled_data_to_grid" that reads ACLED data for the years 1997 to 2019 and creates a geospatial grid stack of reported fatalities or conflict events, where each grid has the totals for a given month and year. It uses the latitude and longitude associated with each conflict to place it in the appropriate grid cell. ACLED conflicts may be reported for specific locations, cities, or administrative zones of various sizes, but in all cases the recorded latitude and longitude is for the centroid of that region. Because of this, it is unfortunately not possible to capture the actual geospatial extent of the conflict for comparison to results from the Stochastic Conflict Model. In addition, the total number of fatalities associated with a conflict are spread evenly over the duration of the conflict. For example, the estimated number of deaths in the Ethiopia-Eritrea war is 100,000, but they occurred in a broad region near the border town of Badme over a 73-day period, so there are 73 ACLED entries (rows) that report 1369 deaths (100,000/73). The figure below shows one of the images created by applying the acled.py utility to the geographic bounding box of the Greater Horn of Africa.

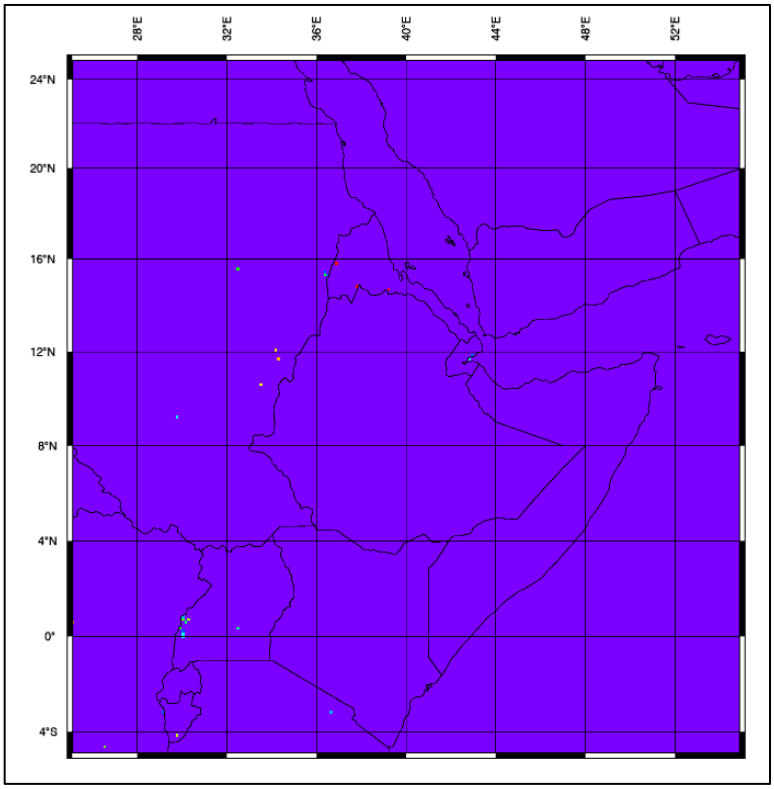


Figure 12. ACLED conflicts (collapsed to points) for one day during the Ethiopia-Eritrea war. Red dots near the center are close to the contested border town of Badme. Other dots can be seen near the border between Uganda and Democratic Republic of the Congo, and the border between Ethiopia and Sudan/South Sudan.

Preparing Input Grids for the Model

The Stochastic Conflict Model is set up to accept geospatial grids of unrest (U_file), local connectivity (C1_file) and nonlocal connectivity (C2_file) in GeoTIFF format. Unfortunately, it is difficult to find suitable gridded data for most indicators of interest. Virtually all available data sets are provided as spreadsheets for administrative zones as opposed to geospatial grids, and therefore have a fairly coarse spatial resolution. Those that are available as geospatial grids were derived from data for administrative zones via spatial interpolation and have similarly low spatial resolution. Another issue is that when higher-resolution data sets are available, they are often either proprietary or have restricted access due to privacy or security concerns. For use in the model, gridded data must be converted to GeoTIFF format. This can be done using widely available GIS tools. In Python, for example, the well-known ***gdal*** package can be used. Here we mention a few of the most promising data sets that are available.

Gridded Population of the World, Version 4 (NASA). This global, gridded data set was created by applying a spatial interpolation algorithm to population count data (people per grid cell) for sub-national administrative zones. While it is available with a spatial resolution of 30 arc-seconds (roughly 1 km near the equator), and can easily be aggregated to coarser resolutions, the actual, much coarser spatial resolution is readily apparent and administrative boundaries are clearly visible. It is available for the following years: 2005, 2010, 2015, and 2020. The 2020 data is used as the default for both the unrest and local connectivity grids. The Stochastic Conflict Model package includes a module: conflict/utils/geotiff.py that provides a function called "regrid_geotiff()" that can be used to create grids for the model region at different resolutions. More information on this data set is available at:
https://cmr.earthdata.nasa.gov/search/concepts/C1597158029-SEDAC.html.
Note that this data set has large nodata regions for the country of Egypt, which affects the top part of the model's default domain. This issue is explained at:
https://sedac.uservoice.com/knowledgebase/articles/799203-why-does-egypt-look-different-than-the-rest-of-nor.

Inverse Distance to Own Country's Border. The Stochastic Conflict Model package includes a module: conflict/utils/distance_to_border.py that uses a grid of ISO country codes at 0.1-degree resolution (360 arcseconds, roughly 11.1 km) to create two grids: Distance to own border and Inverse distance to own border. The inverse distance is defined as: d_inv = exp(-6 * d), so that it is 1 for grid cells on a country's border (excluding ocean borders) and decreases rapidly with distance away from the border. Conflicts between nations often occur along their common border and near population centers. This grid can therefore be used in the determination of the Unrest grid, U. See Figure 13, and Figures 10a, 10b, 10c, and 11a in the Examples section.

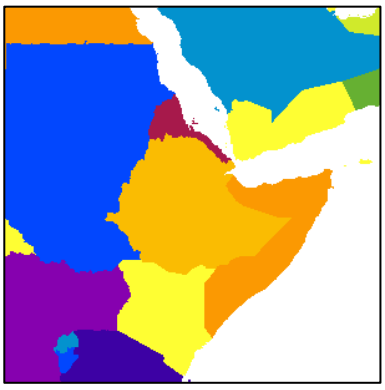
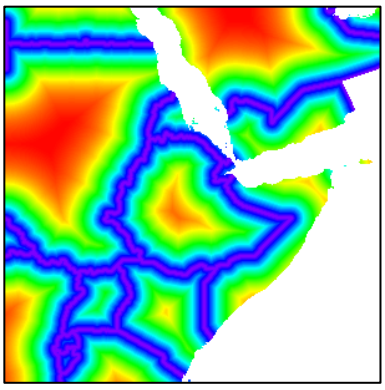
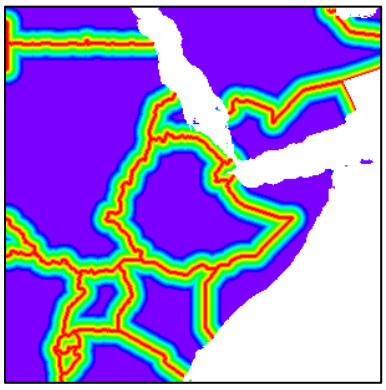
a.  b.  c. 

Figure 13. (a) Color image of ISO country codes. (b) Distance to own country's border, d. (c) Inverse distance to own country's border: exp(-6 * d). In (b) and (c), high values are red and low values are blue and purple. Distance and inverse distance computed by the module: conflict/utils/distance_to_border.py.

PRIO-GRID 2.0. While this data set provides global, gridded data for numerous indicator variables and a range of years, the spatial resolution is only half a degree (30 arcminutes) or 1800 arcseconds. So, the entire global grid has only 720 columns (longitude) and 360 rows (latitude). While this data set includes the indicator "Distance to own borders", it has a much lower spatial resolution (0.5 degree) than what can be computed with the utility described in the last section (0.1 degree). Many other indicators of interest are included, such as: Precipitation total, Proportion of main crop growing season experience drought, Nightlights (mean), and Travel time to nearest urban center (average), Child malnutrition (average), and Infant mortality rate (average). Population count data from GPW-v4 is also included. See: https://grid.prio.org/ and Tollefsen et al. (2012).

GeoIP Data (Internet Access). This interesting global data set provides geolocation metadata (e.g., latitude and longitude) for registered IP addresses worldwide. It includes both the older IPv4 and newer IPv6 addresses. While a coarser-resolution version of this data set (GeoLite) is available for free, the high-resolution data must be purchased from MaxMind, the company that produced it. This data set could be used to create a geospatial grid for the indicator "Internet access", which could then be used as the nonlocal connectivity grid (C2_file) for the model. Note, however, that the geolocation associated with an IP address is often that of the internet service provider, as opposed to that of the household (end-user). While internet access and use in Africa is growing steadily, most people still do not have access.
https://www.maxmind.com/en/geoip2-services-and-databases.
https://www.r-bloggers.com/2014/09/mapping-every-ipv4-address/.

Wealth/Poverty Data as a Surrogate for Internet Access. It has been shown by a Pew Research Center (PWC, 2016) study that there is a high correlation (0.87) between internet access and wealth. Lee and Braithwaite (2009) have developed a machine-learning method for producing high-resolution poverty/wealth maps and applied it to 25 sub-Saharan Africa. It may therefore be possible to estimate internet access (on a geospatial grid) by using geospatial poverty maps. The resulting grid could then be used for the nonlocal connectivity grid (C2_file) in the model.

Conclusions and Future Work

The Stochastic Conflict Model simulates the three fundamental processes associated with conflicts --- namely emergence, spread, and resolution --- as a stochastic, spatial process. It has been set up in such a way that the probabilities associated with each process can vary geospatially and in time by providing the model with gridded data for unrest, local connectivity, and nonlocal connectivity. It can be shown that the model is a type of Markov process, which is a type of stochastic process where the next state of the system depends only on the previous state. The special case of no spreading (only emergence and resolution) is mathematically tractable and analytic results for this case are given in Appendix 3. Simulation results match theoretical predictions for this special case, which provides assurance that the model has been implemented correctly. Analytical (closed-form) results for the general case with spreading (either local or nonlocal) are not yet available. However, both visual assessment of simulations and results from Markov theory suggest that in most cases the model evolves from its initial condition to a statistical steady-state (invariant distribution) condition. Visually, this means that for any fixed set of model parameters, a particular spatial pattern emerges that is typically different than what is observed for another fixed set of parameters. The spatial patterns differ in various ways, including the average size of the conflict clusters, the separation distance between clusters and the appearance of areas without conflict (i.e., voids). The wide variety of possible patterns was illustrated in the section titled Examples for the Greater Horn of Africa. Of course, some of these patterns are expected to differ considerably from what is observed, but they illustrate the breadth of scenarios that the model can simulate.

The model is quite fast, even for relatively large spatial grids. This is due to the use of a Python package called numpy (Numerical Python), careful vectorization of all computations with numpy commands to avoid spatial for loops and the fact that model writes its output efficiently to files in binary format. The model uses random numbers from a Bernoulli distribution, using numpy's random number generator module, numpy.random.

The model is very easy to use, particularly when using the Jupyter notebook GUI that is included with the Python package. It can also be run at an OS terminal prompt or a Python prompt. While the GUI is currently hard-wired to run the model for the Greater Horn of Africa region, it is easily adapted to run for any geographic bounding box by changing parameters in its configuration file.

The model can be run in a Docker container, and this was demonstrated using the Dojo software developed by Jataware. Results from Dojo, including detailed annotation of inputs and outputs, were successfully imported into Causemos, developed by Uncharted. Appendix 2 provides complete details on setting up and running the model in a Docker container with Dojo.

Despite very encouraging results, the full potential of the model has not yet been realized due to the difficulty associated with obtaining high-resolution, gridded data for:

(1) relevant indicators for generating the input grids (U_file, C1_file, C2_file) and
(2) actual, geospatial conflict data for calibrating and verifying the model.

Future work will focus on addressing these issues, by pursuing the ideas described in the section titled: Preparing Input Grids for the Model. It is also possible to use output from machine-learning models (such as those developed by Kimetrica) for generating grids of unrest, local connectivity, and nonlocal connectivity.

Appendix 1: The Forest-Fire Model

Classic versions of the forest-fire model were introduced by Bak et al. (1990), Chen et al. (1990), and Drossel and Schwabl (1992). The model is executed on a spatial grid, usually 2D and rectilinear. Each grid cell has 3 possible states: empty, occupied by a tree, or burning. These 3 states can be represented by 0, 1, and 2, respectively. Initial values can be assigned in a variety of different ways, such as all empty, all trees, etc. During any timestep, grid cells transition between these 3 possible states according to 4 simple rules:

(1) **Triggering** (1 to 2). A "tree cell" with no burning neighbors catches fire with probability f. (Think of a lightning strike.)
(2) **Spreading** (1 to 2). A "tree cell" with at least one burning neighbor (4 nearest) catches fire.
(3) **Burnout** (2 to 0). A burning cell turns into an empty cell; it burns for one timestep.
(4) **Regrowth** (0 to 1). An empty cell becomes a "tree cell" with probability, p.

It has been shown that the behavior of this model is controlled by the ratio: p/f, which gives the average number of trees that grow between two successive lightning strikes. If f << p, large clusters of trees can develop. Under certain conditions, the distribution of cluster sizes exhibit scaling or fractal behavior (i.e., a power-law distribution). See Grassberger (2002).

There are a few similarities, but many differences, between this forest-fire model and the Stochastic Conflict Model (SCM). The main differences are:

(1) In the forest-fire model, there is no distinction between different grid cells; they all have the same probabilities. In the SCM, probabilities are determined by values in the 3 geospatial input grids, U, C1 and C2.
(2) In the forest-fire model, there is only local spreading. Nonlocal spreading cannot occur.
(3) In the forest-fire model, trees only burn for one time step. In the SCM, conflicts can persist over many timesteps until they are eventually resolved.
(4) In the forest-fire model, a grid cell can have 3 possible states. While the SCM also has 3 possible states (conflict, no-conflict and "off-limits"), the off-limits grid cells do not participate in the process. Off-limits grid cells include unpopulated areas, such as oceans and lakes.

Appendix 2: Containerization of the Model with Dojo

Review the documentation for Dojo:

Read: https://docs.dojo-test.com/model-registration.html

Read: https://docs.dojo-test.com/model-registration.html#container-set-up

Set up a model execution container with Dojo:

(1) Go to: https://phantom.dojo-test.com.
(2) Log in with: user: wmuser, pwd: sunsetmoth
(3) Click the Go button for "A Model".
(4) Provide all registration information.
(5) Choose a worker.
(6) Choose "Ubuntu" as the base image.

Clone the model repository

Let "$" denote "/home/clouseau$".

$ git clone https://github.com/peckhams/stochastic_conflict_model

```
Cloning into 'stochastic_conflict_model'...
remote: Enumerating objects: 80, done.
remote: Counting objects: 100% (80/80), done.
remote: Compressing objects: 100% (72/72), done.
remote: Total 80 (delta 29), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (80/80), 348.65 KiB | 7.42 MiB/s, done.
```

Install miniconda for Linux

See: https://gist.github.com/arose13/fcc1d2d5ad67503ba9842ea64f6bac35
Note: repo.continuum.io is now repo.anaconda.com.

```
$ wget https://repo.anaconda.com/miniconda/Miniconda3-latest-Linux-x86_64.sh -O ~/miniconda.sh
$ bash ~/miniconda.sh -b -p ~/miniconda
$ rm ~/miniconda.sh
$ export PATH=~/miniconda/bin:$PATH
$ which conda
    /home/clouseau/miniconda/bin/conda
```

The "export" command above can be added to ~/.bashrc or repeated at the beginning of each session.

Create directory for model output files

```
$ cd
$ mkdir output
```

Install required Python packages with conda

$ conda update -n base conda

$ conda install numpy

$ conda install pandas

$ conda install matplotlib

$ conda install gdal

$ conda install netCDF4

$ conda install imageio

$ conda install pip

$ pip install imageio-ffmpeg (an imageio extension)

Install the model package using its setup.py

Note that the second command here includes a dot "." at the end.

$ cd stochastic_conflict_model

$ pip install -e .

Copy files into the Dojo container (if necessary)

You can copy files into the Dojo container as follows.
(1) First create a folder hierarchy with the data.
(2) Compress the folder hierarchy to create zip file.
(3) Place the zip file in your local Dropbox folder.
(4) Right click on the zip file and choose "Copy Dropbox link"
(5) Finally, type the following commands:

$ which wget

/usr/bin/wget

$ cd some_dir

$ wget -r -O files.zip <dropbox_link>

$ unzip files.zip

Run the model with default CFG file

$ python conflict --cfg_file './input_files/conflict.cfg'

Run the model with another CFG file

$ python conflict --cfg_file './input_files/conflict_Upopcount1.cfg'

Run the model with another CFG file

$ python conflict --cfg_file './input_files/conflict_Upopcount2.cfg'

Appendix 3: Analytic Results for the Case of No Spreading

Consider a grid cell at location (i,j). Let p(k) = p(i,j,k), which depends on U(k) = U(i,j,k), and let B1(k) be a sequence of independent Bernoulli random variables with parameter p(k). Let B2(k) be a sequence of independent Bernoulli random variables with parameter p_resolve. In order to have S(k+1)=0, there are only two (mutually exclusive) possibilities:

(1) [S(k) = 0] and [no conflict was triggered at time k+1], or
(2) [S(k) = 1] and [the conflict was resolved at time k+1].

Recall that for two ***independent*** random events X and Y, the probability of their intersection is given by: P[X and Y] = P[X] P[Y]. (A random event can be that some random variable takes a particular value.) Notice that B1(k+1) and B2(k+1) have no dependence on S(k). Note also that for two ***mutually exclusive*** events X and Y, P(X and Y) = 0. (e.g. S=0 and S=1). Also recall that the probability of the union of events X and Y is given by: P[X or Y] = P(X) + P(Y) - P(X and Y). (This one does not require independence.) The probability that S(k+1)=0 can therefore be computed as follows:

P[S(k+1) = 0] = P[{(S(k) = 0) and (B1(k+1) = 0)} OR {(S(k) = 1) and (B2(k+1) = 1}]
= P[S(k) = 0] P[B1(k+1) = 0] + P[S(k) = 1] P[B2(k+1) = 1]

P[S(k+1) = 1] = 1 - P[S(k+1) = 0]

Let R(k) = P[S(k) = 0], let a(k) = pe(k) = p_emerge(k) = p(i,j,k), and let b = pr = p_resolve. We then have the equation:

R(k+1) = R(k)*[1 - pe(k)] + [1 - R(k)]*pr
R(k+1) = pr + R(k) * [1 - pe(k) - pr]
R(k+1) = R(k) * a(k) + b.

where b = pr and a(k) = [1 - pe(k) - pr]. Note that R(0) = 1 since S(0)=0 (i.e. S is initialized to 0). This is a ***recurrence equation*** for R(k). In order to solve it, we must know a(k), which requires knowing both pe(k) and U(k). However, in the special case where the unrest, U, is the same for every time interval, k --- so that U(k) = U(0) --- we have a(k) = a(0) = a = f(U(0)) (i.e. a function of U(0)). In this special case (still with no spreading) we can solve the recurrence equation for R(k) (with Mathematica's RSolve function), to get

R(k) = {{a^k * (a + b - 1) - b } / (a - 1)}.

Since a = (1 - pe - pr) and b = pr, (a + b - 1) = -pe, and (a - 1) = -(pe + pr). After these substitutions we can simplify to get

P[S(k) = 0] = R(k) = [pr + pe *(1 - pe - pr)^k] / (pr + pe)

Notice that the term (1 - pe - pr) will be negative if (pe + pr) > 1 (e.g. pe = pr = 0.6). While the equation is still valid in this case, (1 - pe - pr)^k will be negative for odd values of k and positive otherwise; this will cause R(k) to oscillate rather than decreasing smoothly to its limiting value. In the limit as k goes to infinity, this sequence rapidly converges to the constant:

$$c = pr / (pe + pr) < 1.$$

If pe = pr, this constant equals 0.5. Keep in mind that pr is a constant (i.e. it does not depend on i, j or k). For this result we assumed that U(i,j,k) is the same for all k, but this means that pe can still be different for every grid cell, that is, pe = pe(i,j). Note that if pe << pr, then c approaches 1 and the given cell will be very ***unlikely*** to have conflict. Similarly, if pe >> pr, then c approaches 0 and the given cell is very ***likely*** to have conflict. If U does not depend on i, j or k, then pe won't either.

Note: If U is spatially uniform and c_spread = c_spread2 = 0 (or C1 = C2 = 0), this result also implies that after sufficiently many timesteps, the fraction of **all** grid cells that have S(k)=0 should converge to: pr/(pe + pr) and the fraction that have S(k)=1 should converge to: pe/(pe+pr). This prediction closely matches the model results. The fraction is reported in the console window at the end of each model run. The model will use spatially uniform U if U_file is left blank.

Formulation as a Two-State Markov Process

The model can be formulated as a 2-state Markov process, which allows the results of the last section to instead be obtained by using results from the theory of Markov processes (see Markov chain, 2022). For the grid cell at (i,j), S(i,j,k) = S(k) is a sequence of zeros and ones (the two possible states -- conflict or no conflict). (The special value S(k)=2 is only used to exclude cells, as in the ocean.)

P[S(k+1) = 1 | S(k) = 0] = P[B1(k+1) = 1] = pe (new conflict emerges)
P[S(k+1) = 0 | S(k) = 0] = P[B1(k+1) = 0] = (1 - pe) (current non-conflict continues)
P[S(k+1) = 1 | S(k) = 1] = P[B2(k+1) = 0] = (1 - pr) (current conflict continues)
P[S(k+1) = 0 | S(k) = 1] = P[B2(k+1) = 1] = pr (current conflict is resolved)

These probabilities give the transition matrix for a two-state Markov process:

M = [[(1 - pe), pe], = [[0 to 0, 0 to 1]
[pr, (1 - pr)]] [1 to 0, 1 to 1]]

But Markov theory says that the probability of going from any state to another state in k steps is given by M^k, the kth power of the transition matrix, M. Combining this with the fact that:

$$P[S(k+1) = 0] = P[S(k) = 0] P[B1(k+1) = 0] + P[S(k) = 1] P[B2(k+1) = 1]$$

we recover the main result of the previous section, namely:

$$P[S(k) = 0] = R(k) = [pr + pe *(1 - pe - pr)^k] / (pr + pe).$$

References – Forest-Fire Model

Bak, Per; Chen, Kan; Tang, Chao (1990) A forest-fire model and some thoughts on turbulence. Physics Letters A. Elsevier BV. 147 (5–6): 297–300. ISSN 0375-9601. https://doi.org/10.1016/0375-9601(90)90451-s.

Chen, Kan; Bak, Per; Jensen, Mogens H. (1990) A deterministic critical forest fire model. Physics Letters A. Elsevier BV. 149 (4): 207–210. ISSN 0375-9601. https://doi.org/10.1016/0375-9601(90)90328-l.

Drossel, B. and F. Schwabl (1992) Self-organized critical forest-fire model, Physical Review Letters, Vol. 69, No. 11.

Forest-fire Model (2021), Wikipedia, https://en.wikipedia.org/wiki/Forest-fire_model, Accessed: May 12, 2021

Grassberger, P. (2002) Critical behaviour of the Drossel-Schwabl forest fire model. New Journal of Physics. IOP Publishing. 4: 17–17. doi:10.1088/1367-2630/4/1/317. ISSN 1367-2630.

References – Conflict Models and Data

Boschee, E., J. Lautenschlager, S. O'Brien, S. Shellman, J. Starz (2018) ICEWS Weekly Event Data, https://doi.org/10.7910/DVN/QI2T9A, Harvard Dataverse, V278

Boschee, E., J. Lautenschlager, S. O'Brien, S. Shellman, J. Starz, M. Ward (2015) ICEWS Coded Event Data, https://doi.org/10.7910/DVN/28075, Harvard Dataverse, V30, UNF:6:NOSHB7wyt0SQ8sMg7+w38w== [fileUNF]

Boschee, E., J. Lautenschlager, S. Shellman, A. Shilliday (2015) ICEWS Dictionaries, https://doi.org/10.7910/DVN/28118, Harvard Dataverse, V4, UNF:6:WBLzLxVC+rRZzmsx/wwtHA== [fileUNF]

Clifford, P. and A. Sudbury (1973) A model for spatial conflict, Biometrika, Vol. 60, No. 3, pp. 581-588, Oxford University Press, https://www.jstor.org/stable/2335008

GDELT Project (2021) A global database of events, https://www.gdeltproject.org/

Goldstone, J.A. (2002) Population and security: How demographic change can lead to violent conflict, Journal of International Affairs, Vol. 56, No. 1, 20 pp.

Hadley, L, (2019) Borders and the feasibility of rebel conflict, Borders in Globalization Review, Vol. 1, No. 1, pp. 66-82, https://doi.org/10.18357/bigr11201919259

Halkia, M., S. Ferri, M. Papazoglou, M-S. van Damme, G. Jenkinson, K-M. Baumann, D. Thomakos (2019) Dynamic global conflict risk index, JRC Technical Reports, European Commission, Publications Office of the EU, https://doi.org/10.2760/846412.

Halkia, M., S. Ferri, M. Papazoglou, M-S. van Damme, D. Thomakos (2020) Conflict event modelling: Research experiment and event data limitations, Conference: LREC 2020 Workshop on Automated Event Extraction of Socio-political Events from News (AESPEN2020), Marseille, France.
https://www.researchgate.net/publication/341378782_Conflict_Event_Modelling_Research_Experiment_and_Event_Data_Limitations

Hamblin, R.L., M. Hout, J.L.L. Miller, B.L. Pitcher (1977) Arms races: A test of two models, American Sociological Review, Vol. 42, No. 2 (April 1977), pp. 338-354, https://doi.org/10.2307/2094609

Kim, M., K., D. Paini, and R. Jurdak (2019) Modeling stochastic processes in disease spread across a heterogeneous social system, Proceedings of the National Academy of Sciences, Vol. 116, No. 2, 401-406, https://doi.org/10.1073/pnas.1801429116.

Lee, E.D., B.C. Daniels, C.R. Myers, D.C. Krakauer, and J.C. Flack (2020a) Scaling theory of armed-conflict avalanches, Physical Review E, Vol. 102, No. 042312, 12 pp., http://doi.org/10.1103/PhysRevE.102.042312

Lee, E.D., B.C. Daniels, C.R. Myers, D.C. Krakauer, and J.C. Flack (2020b) Emergent regularities and scaling in armed conflict data, arXiv:1903.07762v4 [physics.soc-ph].

Lee, K. and J. Braithwaite (2020) High-resolution poverty maps in sub-Saharan Africa. https://arxiv.org/abs/2009.00544

Morrow, J.D. (1986) A spatial model of international conflict, The American Political Science Review, Vol. 80, No. 4 (Dec. 1986), pp. 1131-1150, American Political Science Association, Stable URL: https://www.jstor.org/stable/1960860.

Nailufar, B., S. Wijaya, and D. Perwitasari (2015) Landscape modeling for human - Sulawesi crested black macaques conflict in North Sulawesi, Procedia Environmental Sciences, 24, pp. 104-110. http://doi.org/10.1016/j.proenv.2015.03.014. (example of logistic regression)

PWC (2016) Pew Research Center report, https://www.pewresearch.org/global/2016/02/22/internet-access-growing-worldwide-but-remains-higher-in-advanced-economies/.

Rustad, S.A., H. Buhaug, Å. Falch, and S. Gates (2011) All conflict is local: Modeling subnational variation in civil conflict risk, Conflict Management and Peace Science, Vol. 28, No. 1, pp. 15-40.

Tollefsen, A.F., H. Strand and H. Buhaug (2012) PRIO-GRID: A unified spatial data structure, Journal of Peace Research, 49(2), 363-374.

UCDP (2021) Uppsala Conflict Data Program, https://www.pcr.uu.se/research/ucdp/.

USAID (2012) Conflict Assessment Framework: Application Guide, United States Agency for International Development. 48 pp., https://pdf.usaid.gov/pdf_docs/PNADY740.pdf

ViEWS (2021) A political Violence Early Warning System, Dept. of Peace and Conflict Research, Uppsala University, https://www.pcr.uu.se/research/views/.

Walther, O.J., S.M. Radil, D.G. Russell, and M. Trémolières (2020) Introducing the Spatial Conflict Dynamics indicator of political violence, arXiv:2003.01750v2 [physics.soc-ph].

Yurevich, P.A., M.A. Olegovich, S.V. Mikhailovich, and P.Y. Vasilievich (2018) Modeling conflict in a social system using diffusion equations, Simulation: Transactions of the Society for Modeling and Simulation International, Vol. 94, No. 12, pp. 1053-1061, http://doi.org/10.1177/0037549718761573.

References – Wikipedia

Arab Spring (2022) Wikipedia, https://en.wikipedia.org/wiki/Arab_Spring, Accessed: Jan. 15, 2022

Asynchronous Cellular Automaton (2022), Wikipedia, (see random independent update scheme) https://en.wikipedia.org/wiki/Asynchronous_cellular_automaton, Jan. 15, 2022

Bernoulli Distribution (2022), Wikipedia, https://en.wikipedia.org/wiki/Bernoulli_distribution, Accessed: Jan. 15, 2022

Binary Regression (2022), Wikipedia, https://en.wikipedia.org/wiki/Binary_regression, Accessed: Jan. 15, 2022

Binomial Distribution (2022), Wikipedia, https://en.wikipedia.org/wiki/Binomial_distribution, Accessed: Jan.15, 2022

Cellular Automaton (2022), Wikipedia, https://en.wikipedia.org/wiki/Cellular_automaton, Accessed: Jan. 15, 2022

Combustibility and Flammability (2022) Wikipedia, https://en.wikipedia.org/wiki/Combustibility_and_flammability, Accessed: Jan. 15, 2022

Compartmental Models (2022) Compartmental models in epidemiology, Wikipedia, https://en.wikipedia.org/wiki/Compartmental_models_in_epidemiology, Accessed: Jan. 15, 2022

Exponential Distribution (2022) Wikipedia, https://en.wikipedia.org/wiki/Exponential_distribution, Accessed: Jan. 15, 2022

Forest-fire Model (2022), Wikipedia, https://en.wikipedia.org/wiki/Forest-fire_model, Accessed: Jan. 15, 2022

Geometric Distribution (2022) Wikipedia, https://en.wikipedia.org/wiki/Geometric_distribution, Accessed: Jan. 15, 2022

Hatfield and McCoy Feud (2022), Wikipedia, https://en.wikipedia.org/wiki/Hatfield%E2%80%93McCoy_feud, Accessed: Jan. 15, 2022, classic example of self-perpetuated conflict.

Infectious Disease Models (2022), Mathematical modeling of infectious disease, Wikipedia, https://en.wikipedia.org/wiki/Mathematical_modelling_of_infectious_disease, Accessed: Jan. 15, 2022

Logistic Regression (2022), Wikipedia, https://en.wikipedia.org/wiki/Logistic_regression, Accessed: Jan. 15, 2022

Markov Chain (2022) Wikipedia, https://en.wikipedia.org/wiki/Markov_chain, Accessed: Jan. 15, 2022