disturbanceGenerator_NT

Tati Micheletti

22 August 2022

Overview

The `disturbanceGenerator_NT` is a parent module for three children modules that are stand alone: `anthroDisturbance_DataPrep`, `potentialResourcesNT_DataPrep` and `anthroDisturbance_Generator`, the second being idiosyncratic to Northwest Territories data. The other two modules are, however, generic and can be applied in other contexts, potentially without modifications.

These modules aim at simulating anthropogenic disturbance generation (i.e., mining, oil and gas, forestry, etc.) and were primarily developed for improving caribou range planning in Northern Canada. Each module is detailed below. In this report, we provide an example of usage of the three modules together. Each individual module, however, can be used used stand alone and references on how to use them can be found in their individual documentation.

anthroDisturbance_DataPrep

This is a data preparation module to harmonize different anthropogenic disturbance datasets. It's primarily intended for the Northwest Territories region (default), but the structure is universal. All needed is the metadata information required by the `disturbanceDT` object (a `data.table`) and it generates the list (general category) of lists (specific class) needed for generating disturbance layers.

Parameters

This module has three parameters that can be adjusted by the user. Namely:

- `whatNotToCombine`: This is a character string and defaults to `potential`. Here the user should specify which `dataClass` from the object `disturbances` should NOT be combined. This is especially important for potential resource layers that need idiosyncratic processes to be generated, which might happen on a separate module (i.e., `potentialResourcesNT_DataPrep`). This parameter is used as a pattern string to identify which `dataClass` contains the pattern and excludes these from the harmonization of the datasets;
- `useSavedList`: This is a logical and defaults to `TRUE`. If the `disturbances` object was saved by a previous run of the module and this parameter is `TRUE`, it returns the saved list. This might save a considerable amount of time but attention is needed to make sure the saved objects are indeed the expected ones;
- `skipFixErrors`: This is a character string and defaults to a couple of layers used by the default objects (i.e., NT_FORCOV.shp, NorthwestTerritories_15m_Disturb_Perturb_Poly.shp, polygonalDisturbances_NT1_BCR6_clipped.shp, NorthwestTerritories_15m_Disturb_Perturb_Line.shp, linearDisturbances_NT1_BCR6_clipped.shp). Some polygons have errors that are automatically

fixed in the function `createDistubanceList()`. However, some polygons are fine and too big to be checked. Here you can pass the name of the files you believe are correct, but would take a long time to be checked. This will skip the function `fixErrors()` for these files and improve module speed.

Inputs

The module expects only one input. The `disturbanceDT` data table contains the following columns:

- `dataName`: this column groups the type of data by sector (i.e., Energy, Settlements, OilGas, Mining, Forestry, Roads);
- `URL`: URL link for the specific dataset;
- `classToSearch`: exact polygon type/class to search for when picking from a dataset with multiple types. If this is not used (i.e., your shapefile is alreday all the data needed), you should still specify this so each entry has a different name;
- `fieldToSearch`: where should classToSearch be found? If this is specified, then the function will subset the spatial object (most likely a shapefile) to classToSearch. Only provide this if this is necessary!
- `dataClass`: this column details the type of data further (i.e., Settlements, potentialSettlements otherPolygons, otherLines, windTurbines, potentialWindTurbines, hydroStations, oilFacilities, pipelines, etc). Common class to rename the dataset to, so we can harmonize different ones. Potential data classes can be of three general types (that will be specified in the disturbanceGenerator module as a parameter – ALWAYS with 'potential' starting):
  - Enlarging (i.e., potentialSettlements and potentialSeismicLines): where the potential one is exactly the same as the current layer, and we only buffer it with time;
  - Generating (i.e., potentialWind, potentialOilGas, potentialMineral, potentialCutblocks): where the potential layers are only the potential where structures can appear based on a specific rate;
  - Connecting (i.e., potentialPipelines, potentialTransmission, potentialRoads incl. forestry ones): where the potential layer needs to have the current/latest transmission, pipeline, and road network. This process will depend on what is generated in point 2.;
- `fileName`: If the original file is a .zip and the features are stored in one of more shapefiles inside the .zip, please provide which shapefile to be used;
- `dataType`: Provide the data type of the layer to be used. These are the current accepted formats: 'shapefile' (.shp or .gdb), 'raster' (.tif, which will be converted into shapefile), and 'mif' (which will be read as a shapefile). The last defaults to an example in the Northwest Territories and needs to be provided if the study area is not in this region (i.e., union of BCR6 and NT1).
  Other inputs needed by this module are the the study area (`studyArea`) and a raster ('rasterToMatch") that matches the study area and provides the spatial resolution for the simulations.

Outputs

The module outputs the following objects:

- `disturbances`: List (general category or sector) of lists (specific class or sub-sector) of spatial elements (i.e., rasters or shapefiles) needed for generating disturbances. The specific classes' are structured in the following way:

  - Outter list's names: match the `dataName` from `disturbanceDT`;

  - Inner list's names: match the `dataClass` from `disturbanceDT`, which might not be a unique class.

- `disturbanceList`: List (general category or sector) of lists (specific class or sub-sector) of spatial elements (i.e., rasters or shapefiles) needed for generating disturbances. The specific classes' are structured in the following way:
  - Outter list's names: `dataName` from `disturbanceDT`;

  - Inner list's names: `dataClass` from `disturbanceDT`, which is a unique class after harmozining, except for any potential resources that need idiosyncratic processing. This means that each combination of `dataName` and `dataClass` (except for 'potential') will have only one element. Another module can deal with the potential layers. For the current defaults, this is the `potentialResourcesNT_DataPrep`. If none of the potential layers needs to be modified or combined, you might skip this idiosyncratic module and directly use the `anthroDisturbance_Generator` module.

Events

This module contains only one event, named `loadAndHarmonizeDisturbanceDT`. This event has three main steps:

1. Create the disturbance list: the function `createDisturbanceList()` loops over `dataName`, followed by `dataClass`, and `classToSearch` and

- downloads the necessary data,
- place this data in the module's data folder,
- loads the layer,
- reprojects any non-matching layers to the rasterToMatch's projection,
- crops to study area;
- once out the `classToSearch` loop, it merge or unifies the layers to a common `dataClass`; and
- returns all layers without other union.

2. Wrap `terra` objects: the function `wrapTerraList()` wraps and saves the created disturbance list (`disturbances`) as `terra` objects cannot be directly saved as other spatial object formats. The speed increase when using `terra`, however, is worth the additional step.
3. Harmonize the disturbance list: the function `hamononizeList()` is responsible for generating the `disturbanceList` object. This is where the unification of different sub-sectors that can be simulated together occurs.

Defaults

This module can be run without any inputs from the user and will automatically default to an example in the Northwest Territories of Canada (i.e., union of BCR6 and NT1).

potentialResourcesNT_DataPrep

This is a data preparation module to harmonize different anthropogenic disturbance datasets, more specifically, mining and oil/gas. It's intended for the Northwest Territories region (default) and is idiosyncratic.This means this module is NOT generalizable, but can be used as basis for other types of development. The objective is to create one standardized layer for each of the potential resources that has increasing values for most prioritized places (i.e., higher values, more likely structures will appear).

Parameters

This module has one parameter that can be adjusted by the user:
- `whatToCombine`: This is a data.table and defaults to:

```
DT <- data.table::data.table(datasetName = c("oilGas", "oilGas", "mining", "mining"),
                             dataClasses = c("potentialOilGas", "potentialOilGas",
                                             "potentialMining", "potentialMining"),
                             toDifferentiate = c(NA, "C2H4_BCR6_NT1", "CLAIM_STAT", "PERMIT_STA"),
                             activeProcess = c(NA, NA, "CLAIM_STAT", "PERMIT_STA"))
print(DT)
```

```
##    datasetName     dataClasses toDifferentiate activeProcess
## 1:      oilGas potentialOilGas            <NA>          <NA>
## 2:      oilGas potentialOilGas   C2H4_BCR6_NT1          <NA>
## 3:      mining potentialMining      CLAIM_STAT    CLAIM_STAT
## 4:      mining potentialMining      PERMIT_STA    PERMIT_STA
```

Here the user should specify a data.table with the `dataName` and `dataClasses` from the object `disturbanceList` from the module `anthroDisturbance_DataPrep` to be combined. The table also contains a column identifying which to be used to filter active processes for mining (i.e., `CLAIM_STAT` and `PERMIT_STA`). For Oil/Gas, it needs to identify which layer is the potential one (C2H4_BCR6_NT1) and which is used to constrain where oil and gas will be added. For oil and gas, the other potential layer (exploration permits) is used as a starting point to add structures, followed by randomly placing them in the highest values of `C2H4_BCR6_NT1` and going down until the total amount is reached. For mining, `CLAIM_STAT` is the potential exploration, while `PERMIT_STA` are the ones that might become CLAIMS. The most likely values are `CLAIMS` and followed by `PERMITS`;

Inputs

The module expects four inputs:

- `disturbanceList`: List (general category or sector) of lists (specific class or sub-sector) needed for generating disturbances. The sub-sector last list contains:
  - Outter list's names: `dataName` from `disturbanceDT`;
  - Inner list's names: `dataClass` from disturbanceDT, which is a unique class after harmozining, except for any potential resources that need idiosyncratic processing. This means that each combination of `dataName` and `dataClass` (except for 'potential') will only have only one element;
- `historicalFires`: List per `YEAR` of burned polygons. It needs to contain at least the following columns: `YEAR` or `DECADE`. The default layer was created by ENR for the NT.

The other two inputs needed by this module are the the study area (`studyArea`) and a raster (`rasterToMatch`) that matches the study area and provides the spatial resolution for the simulations.

Outputs

The module outputs only one object, the `disturbanceList`, which is a modified version of the input one, where multiple potential layers (i.e., mining and oilGas) have been replaced by only one layer, with the highest values being the ones that need to be filled with new developments first.

Events

This module contains four events, which are similar in objective, but specific in the inputs and outputs. All events (`createPotentialMining`, `createPotentialOilGas`, `createPotentialCutblocks`, and `replaceInDisturbanceList`) aim at working on all current and potential disturbance layers, unifying these, and setting the highest values as the locations that need to be filled with new developments first, or prepare potential layers. An important note is that for the function `makePotentialCutblocks()`, we recreated the steps done by ENR (J. Hodson) to create the potential forest. Specific information can be found as comments in the function.

Defaults

This module can be run without any inputs from the user and will automatically default to an example in the Northwest Territories of Canada (i.e., union of BCR6 and NT1).

anthroDisturbance_Generator

This is a module to generate anthropogenic disturbances. It's primarily intended for the Northwest Territories region (default), but the structure is universal. All needed to do is provide the metadata information required by the `disturbanceParameters` object and the `disturbanceList` containing current disturbances and potential disturbances (i.e., for disturbances of Generating type, explained below).

Parameters

This module has a couple parameters that can be adjusted by the user. Namely:

- `saveInitialDisturbances`: This is a logical and defaults to `TRUE`. Should the disturbance rasters be saved at each step? These are saved to `Paths[['outputPath']]` as a `RasterLayer`, with `disturbanceLayer` as prefix the name of the industry (i.e., Sector) and the year as suffix. If `TRUE`, it saves the initial conditions (IC);
- `saveCurrentDisturbances`: This is a logical and defaults to `TRUE`. Should the disturbance rasters be saved at each step? These are saved to `Paths[['outputPath']]` as a `RasterLayer`, with `disturbanceLayer` as prefix the name of the industry (i.e., Sector) and the year as suffix. If `TRUE`, it saves at the end of each step;
- `useECCCData`: This is a logical and defaults to `TRUE`. If the rate of change (i.e., rate the disturbances will be generated across the landscape) is not provided, the module can either try to extract it from data if `useECCCData = TRUE` (i.e., ECCC human footprint for 2010 and 2015 data), or simply apply a rate of 0.2% (of the total area) increase per year;
- `checkChangeInDisturbance`: This is a logical and defaults to `FALSE`. When `TRUE`, this prints the change in ECCC human footprint (2010 to 2015) at 30m resolution over the shapefile provided. However, this operation is time consuming;

- `checkDisturbance2015`: This is a logical and defaults to `FALSE`. Prints the total % of ECCC human footprint (2010 to 2015) at 500m resolution over the shapefile provided. However, this operation is time consuming;

- `overwriteDisturbanceLayers2015`: This is a logical and defaults to `FALSE`. Should the disturbance layer from 2015 be overwritten? This is handy when the disturbance layer has changed;

- `overwriteDisturbanceLayers2010`: This is a logical and defaults to `FALSE`. Should the disturbance layer from 2010 be overwritten? This is handy when the disturbance layer has changed;

- `growthStepEnlargingPolys`: This is a numeric and defaults to `0.0075`. Growth step used for iteratively achieving the total area growth of new disturbances type Enlarging for polygons. If the iterations take too long, one should increase this number. If the summarized value is too far from 0, one should decrease this number;

- `growthStepEnlargingLines`: This is a numeric and defaults to `0.1`. Growth step used for iteratively achieving the total area growth of new disturbances type Enlarging for lines. If the iterations take too long, one should increase this number. If the summarized value is too far from 0, one should decrease this number.

Inputs

The module expects four inputs:

- `disturbanceList`: List (general category) of lists (specific class) needed for generating disturbances. This is generally the output from a potentialResources module (i.e., potentialResourcesNT_DataPrep), where multiple potential layers (i.e., mining and oilGas) we replaced by only one layer with the highestvalues being the ones that need to be filled with new developments first, or prepared potential layers (i.e., potentialCutblocks);
- `disturbanceParameters`: Table (`data.table`) with the following columns:
  - `dataName`: this column groups the type of data by sector (i.e., Energy, Settlements, OilGas, Mining, Forestry, Roads);
  - `dataClass`: this column details the type of data ALWAYS with 'potential' starting (i.e., potentialSettlements, potentialWindTurbines, potentialCutblocks, etc.) can harmonize different ones. Potential data classes can be of three general `disturbanceType` (see below);
  - `disturbanceType`: Potential data classes can be of three general types:
    1. Enlarging (i.e., potentialSettlements and potentialSeismicLines): where the potential one is exactly the same as the current layer, and we only buffer it with time;

    2. Generating (i.e., potentialWindTurbines, potentialOilGaspotentialMineral, potentialForestry): where the potential layers are only the potential where structures can appear based on a specific rate;

    3. Connecting (i.e., potentialPipelines, potentialTransmission, potentialRoads incl. forestry ones): where the potential layer needs to have the current/latest transmission, pipeline, and road network. This process will depend on what is generated in point 2;

  - `disturbanceRate`: what is the rate of generation for disturbances per year of type Enlarging and Generating. For disturbances type Connecting, disturbanceRate is NA. If not specified when needed, the module will try to derive it from data (i.e., ECCC). If this fails, it will fall on a yearly average of 0.2% of the current disturbance (except for windTurbine, which has a default value of 1 per 10 years considering the reduced of potential in the region;
  - `disturbanceSize`: if there is a specific size the disturbance in m2 type Generating should have, it is specified here. If not specified, the module will try to derive it from data. For disturbances type Enlarging and Connecting, disturbanceSize is NA;
  - `disturbanceOrigin`: `dataClass` that should be used as the 'original' to be either modified (i.e., Enlarging, Generating) or as origin point for Connecting types;
  - `disturbanceEnd`: end points for Connecting layers (i.e., newly created windTurbines: connect into powerLines, newly created windTurbines: connect into roads, newly created oilGas: connect into pipeline, newly created oilGas: connect into roads, newly created settlements: connect into roads, newly created mines: connect into roadsnewly created cutblocks: connect into roads);
  - `disturbanceInterval`: interval for which this disturbance should happen It defaults to an example in the Northwest Territories and needs to be provided if the study area is not in this region (i.e., union of BCR6 and NT1);

- `rstCurrentBurn`: A binary raster with 1 values representing burned pixels. This raster is normally produced by either the module historicFires or a fire simulation module (i.e., fireSense, SCFM, LandMine);

- `disturbanceDT` data table contains the following columns:

  - `dataName`: this column groups the type of data by sector (i.e., Energy, Settlements, OilGas, Mining, Forestry, Roads);
  - `URL`: URL link for the specific dataset;
  - `classToSearch`: exact polygon type/class to search for when picking from a dataset with multiple types. If this is not used (i.e., your shapefile is alreday all the data needed), you should still specify this so each entry has a different name;
  - `fieldToSearch`: where should classToSearch be found? If this is specified, then the function will subset the spatial object (most likely a shapefile) to classToSearch. Only provide this if this is necessary!
  - `dataClass`: this column details the type of data further (i.e., Settlements, potentialSettlements otherPolygons, otherLines, windTurbines, potentialWindTurbines, hydroStations, oilFacilities, pipelines, etc). Common class to rename the dataset to, so we can harmonize different ones. Potential data classes can be of three general types (that will be specified in the disturbanceGenerator module as a parameter – ALWAYS with 'potential' starting):

    1. Enlarging (i.e., potentialSettlements and potentialSeismicLines): where the potential one is exactly the same as the current layer, and we only buffer it with time;

    2. Generating (i.e., potentialWind, potentialOilGas, potentialMineral, potentialCutblocks): where the potential layers are only the potential where structures can appear based on a specific rate;

    3. Connecting (i.e., potentialPipelines, potentialTransmission, potentialRoads incl. forestry ones): where the potential layer needs to have the current/latest transmission, pipeline, and road network. This process will depend on what is generated in point 2.;

  - `fileName`: If the original file is a .zip and the features are stored in one of more shapefiles inside the .zip, please provide which shapefile to be used;

– `dataType`: Provide the data type of the layer to be used. These are the current accepted formats: 'shapefile' (.shp or .gdb), 'raster' (.tif, which will be converted into shapefile), and 'mif' (which will be read as a shapefile). The last defaults to an example in the Northwest Territories and needs to be provided if the study area is not in this region (i.e., union of BCR6 and NT1).
Other inputs needed by this module are the the study area (`studyArea`) and a raster ('rasterToMatch") that matches the study area and provides the spatial resolution for the simulations.

Outputs

The module outputs the following two objects:

- `disturbanceList`: Updated list (general category) of lists (specific class) of disturbances and the potential needed for generating disturbances;
- `currentDisturbanceLayer`: List (per year) of rasters with all current disturbances. Can be used for other purposes but was created to filter potential pixels that already have disturbances to avoid choosing new pixels in existing disturbed ones.

Events

This module contains four main events:

1. `calculatingSize`: This event happens if the specific sizes of disturbances is not passed to the `disturbanceParameters` table. It uses the current disturbance data to extract the average size of each disturbance type as well as its variation to, later, draw from a normal distribution with the calculated mean and deviation, the sizes of simulated disturbances;
2. `calculatingRate`: This event happens if the specific disturbance growth rate (i.e., proportion of the study area that should be added as new disturbance) is not passed to the `disturbanceParameters` table. If the parameter `useECCCData` is `TRUE`, it uses both 2010 and 2015 human footprint datasets to derive the growth rate of each type of disturbance in a 5-year period and derives the proportion of disturbance over the entire study area from these calculations. This rate is then applied in the `generatingDisturbances` event;
3. `generatingDisturbances`: This is the core event of this module, run by the function `generateDisturbances()`. This function first calculates the total study area to be able to interpret the rates of growth in terms of area size and then masks current disturbances so these are not chosen a second time. Then, three separate steps happen: we grow the disturbances classified as `Enlarging` by buffering the existing disturbances. This is useful especially for settlements and seismic lines, as the likelihood that new disturbances of such types happen in adjacent areas of the current ones is high. Besides, simulating new of these disturbance types in new places is very challenging. The second step is to generate the disturbances classified as `Generating`. These are disturbances such as wind turbines, gas and oil facilities and wells, mines and forestry cut blocks. For these disturbances, we have generated maps (rasters) of the locations with highest potential (i.e., with the `potentialResourcesNT_DataPrep` module) and we iteratively select the areas within these maps, using the calculated size for each type until we get as close as possible to the rate of disturbance. How large the iterative steps are is controlled by the parameters `growthStepEnlargingPolys` and `growthStepEnlargingLines`, with a trade off between precision and run time. At last, the third step connects the new generated disturbances to linear features such as roads and power lines. It uses an algorithm to find the shortest path towards these features. Although this mechanism could be improved by adding topographic and landscape constraints, at a landscape level, the current mechanism might be sufficient. At last, the function converts all shapefiles to rasters and merges them in order to update the current disturbances maps so that in the next module iteration, the chosen pixels are not available;

4. `updatingDisturbanceList`: The last event in the module updates the current disturbed layers with the generated ones. It incorporates new roads and power lines to the current ones, adds the generated disturbances to the current ones and replaces the enlarged settlements and seismic lines with the buffered ones.

Defaults

This module can be run without any inputs from the user and will automatically default to an example in the Northwest Territories of Canada (i.e., union of BCR6 and NT1).

Libraries

The current modules have only been tested with a certain combination of libraries and versions. Please note that the library for the project should be used by all three modules (if all three are used). It is suggested the creation of a central (project level, argument `libPaths`) library location for package installation, which can be done with the following function:

```
Require::Require(packageVersionFile = "packageVersions.txt",
                 libPaths = file.path(getwd(),
                                      "projectLibrary"),
                 standAlone = TRUE)
```

To add this library to the other modules, we suggest the creation of an .Rprofile file containing the information on library location to be loaded *before* the start of the session (avoiding package unloading or loading of wrong package versions). The code below can be added to the .Rprofile:

```
Require::setLibPaths("projectLibrary")
```

Usage

```
if(!require("Require")){
    install.packages("Require")
}
library("Require")
Require("googledrive")
Require("SpaDES.core")
Require("data.table")

# Pass your email for authentication (used for non-interactive calls)
googledrive::drive_auth(email = "tati.micheletti@gmail.com")
options(reproducible.useTerra = FALSE) # Workaround while reproducible is not yet fully functional with

# If you load the project, set the directory where your modules are located
moduleDir <- dirname(getwd())

setPaths(modulePath = moduleDir,
         cachePath = checkPath(file.path(getwd(), "cache"),
                               create = TRUE),
```

```
         outputPath = checkPath(file.path(getwd(), "outputs"),
                                create = TRUE),
         inputPath = checkPath(file.path(getwd(), "inputs"),
                               create = TRUE),
         rasterPath = checkPath(file.path(getwd(), "temp_raster"),
                                create = TRUE))

getPaths() # shows where the 4 relevant paths are

times <- list(start = 2011, end = 2091)

parameters <- list(
  #.progress = list(type = "text", interval = 1), # for a progress bar
    # Default values, don't need to be passed but are here as examples
  anthroDisturbance_DataPrep = list(whatNotToCombine = "potential",
                                    useSavedList = TRUE,
                                    skipFixErrors = c("NT_FORCOV.shp",
                       "NorthwestTerritories_15m_Disturb_Perturb_Poly.shp",
                       "polygonalDisturbances_NT1_BCR6_clipped.shp",
                       "NorthwestTerritories_15m_Disturb_Perturb_Line.shp",
                       "linearDisturbances_NT1_BCR6_clipped.shp")),
  potentialResourcesNT_DataPrep = list(whatToCombine = data.table(datasetName = c("oilGas", "oilGas",
                                                                                  "mining",
                                dataClasses = c("potentialOilGas", "potentialOilGas",
                                                "potentialMining", "potentialMining"),
                                toDifferentiate = c(NA, "C2H4_BCR6_NT1",
                                                    "CLAIM_STAT", "PERMIT_STA"),
                                activeProcess = c(NA, NA,
                                                  "CLAIM_STAT", "PERMIT_STA"))),
    anthroDisturbance_Generator = list(saveInitialDisturbances = TRUE,
                                       saveCurrentDisturbances = TRUE,
                                       useECCCData = TRUE,
                                       checkChangeInDisturbance = FALSE,
                                       checkDisturbance2015 = FALSE,
                                       overwriteDisturbanceLayers2015 = FALSE,
                                       overwriteDisturbanceLayers2010 = FALSE,
                                       growthStepEnlargingPolys = 0.0075,
                                       growthStepEnlargingLines = 0.1
                                       )
)
modules <- list("disturbanceGenerator_NT") # Which includes, in order the following
                                           # modules:
                                           # anthroDisturbance_DataPrep,
                                           # potentialResourcesNT_DataPrep,
                                           # anthroDisturbance_Generator
objects <- list()
inputs <- list()
outputs <- list()

disturbanceList <- simInitAndSpades(times = times,
                                    params = parameters,
                                    modules = modules,
                                    objects = objects)
```