anthroDisturbance_Generator

Tati Micheletti

16 July 2022

Overview

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).

Usage

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

# 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"),
         cachePath = checkPath(file.path(moduleDir, "potentialResourcesNT_DataPrep", "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 = 2041)

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_Generator = list(saveInitialDisturbances = TRUE,
                                     saveCurrentDisturbances = TRUE,
                                     useECCCData = TRUE,
                                     checkChangeInDisturbance = FALSE,
                                     checkDisturbance2015 = FALSE,
                                     overwriteDisturbanceLayers2015 = FALSE,
                                     overwriteDisturbanceLayers2010 = FALSE,
                                     growthStepEnlargingPolys = 0.0075,
                                     growthStepEnlargingLines = 0.1
                                     )
)
modules <- list("anthroDisturbance_Generator")
objects <- list()
inputs <- list()
outputs <- list()

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

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.

```
df_params <- SpaDES.core::moduleParams("anthroDisturbance_Generator", moduleDir)
knitr::kable(df_params)
```

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.

Data dependencies

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).

Input data

The module expects only one input. - `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.

```
df_inputs <- SpaDES.core::moduleInputs("anthroDisturbance_Generator", moduleDir)
knitr::kable(df_inputs)
```

Output data

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.

```
df_outputs <- SpaDES.core::moduleOutputs("anthroDisturbance_Generator", moduleDir)
knitr::kable(df_outputs)
```

Links to other modules

This module can be combined primarily with `anthroDisturbance_DataPrep` and `potentialResourcesNT_DataPrep`, the second being idiosyncratic to Northwest Territories data. The other module is, however, generic and can be applied in other contexts, potentially without modifications, as well as the present one. Such module collection can also be used in combination with (a) landscape simulation module(s) (i.e., LandR) and caribou modules (i.e., caribouRSF, caribouRSF_NT, caribouPopGrowth) to improve realism on simulated landscape and caribou RSF and population growth forecasts.