anthroDisturbance_DataPrep

Tati Micheletti

16 June 2022

Overview

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.

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"),
                               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 = 2011)

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"))
  )
modules <- list("anthroDisturbance_DataPrep")
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.

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

- `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.

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.

Data dependencies

Input data

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):
+ 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_DataPrep", moduleDir)
knitr::kable(df_inputs)
```

Output data

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.

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

Links to other modules

This module can be combined primarily with `potentialResourcesNT_DataPrep` and `anthroDisturbance_Generator`, the first 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.