Debugging scfm

Ian Eddy and Alex Chubaty

Updated May 2023

Overview

See `scfm.Rmd` for an overview of the model.

Usage example

```
if (!"Require" %in% rownames(installed.packages())) {
  install.packages("Require", repos = c("https://predictiveecology.r-universe.dev", getOption("repos")))
}
library(Require)

Install(c("PredictiveEcology/reproducible@development",
          "PredictiveEcology/SpaDES.core@development"),
        dependencies = TRUE, standAlone = TRUE, upgrade = FALSE)
Require(c("data.table", "ggplot2",
          "PredictiveEcology/LandR@development",
          "PredictiveEcology/reproducible@development (>= 2.0.4.9000)",
          "PredictiveEcology/scfmutils@development",
          "sf",
          "PredictiveEcology/SpaDES.core@development",
          "terra"), standAlone = TRUE, upgrade = FALSE)
```

```
paths <- list(
  cachePath = file.path("cache_debug"),
  modulePath = file.path("modules"),
  inputPath = file.path("inputs"),
  outputPath = file.path("outputs_debug")
)

options("reproducible.cachePath" = paths$cachePath)
options("reproducible.useMemoise" = FALSE)
options("reproducible.useTerra" = TRUE)
options("reproducible.rasterRead" = "terra::rast") ## default

timeunit <- "year"
times <- list(start = 1, end = 10)
defaultPlotInterval <- 50
defaultInitialSaveTime <- NA #don't be saving nuffink

globCache <- c(FALSE)  #don't cache the init event - it is too prone to false positives
```

```
parameters <- list(
    scfmDiagnostics = list(
    .studyAreaName = "debug"
  ),
  scfmDriver = list(
    targetN = 4000, # default 4000; increase targetN for more robust estimates, longer run-time
    .useCache = globCache,
    .useParallelFireRegimePolys = TRUE
  ),
  scfmEscape = list(
    .useCache = globCache,
    .plotInitialTime = NA
  ),
  scfmIgnition = list(
    .useCache = globCache
  ),
  scfmLandcoverInit = list(
    sliverThreshold = 1e8, ## polygons <100 km2 are merged with closest non-sliver neighbour
    .useCache = globCache,
  ),
  scfmRegime = list(
    fireCause = c("L"),
    .useCache = globCache
  ),
  scfmSpread = list(
    .plotInitialTime = times$start,
    .plotInterval = 20,
    .useCache = globCache
  )
)

modules <- list(
  "scfmLandcoverInit", "scfmRegime", "scfmDriver",
  "scfmIgnition", "scfmEscape","scfmSpread",
  "scfmDiagnostics"
)

targetCRS <- paste("+proj=lcc +lat_1=49 +lat_2=77 +lat_0=0 +lon_0=-95 +x_0=0 +y_0=0",
                   "+datum=NAD83 +units=m +no_defs +ellps=GRS80 +towgs84=0,0,0")

## canonical 'pseudo-random' SA has coords -1209980, 7586865
studyAreaReporting <- terra::vect(cbind(-1209980, 7586865), crs = targetCRS) |>
  SpaDES.tools::randomStudyArea(center = _, size = 10000 * 250 * 30000, seed = 1002) |>
  st_as_sf()
studyArea <- st_buffer(studyAreaReporting, 5000)
studyAreaLarge <- st_buffer(studyArea, 20000)

rasterToMatchLarge <- rast(ext(studyAreaLarge), res = c(250, 250))
crs(rasterToMatchLarge) <- crs(studyAreaLarge)
rasterToMatchLarge[] <- 1
rasterToMatchLarge <- mask(rasterToMatchLarge, studyAreaLarge)

rasterToMatch <- crop(rasterToMatchLarge, studyArea)
```

```
rasterToMatch <- mask(rasterToMatch, studyArea)

if (FALSE) {
  paths[["modulePath"]] <- "modules/scfm/modules"

  Require::Require("bcdata")
  studyAreaReporting <- bcdata::bcdc_get_data("0bc73892-e41f-41d0-8d8e-828c16139337") |>
    subset(REGION_ORG_UNIT_NAME == "Cariboo Natural Resource Region") |>
    st_union() |>
    st_cast("MULTIPOLYGON")
  studyArea <- st_buffer(studyAreaReporting, 7000)
  studyAreaLarge <- st_buffer(studyArea, 1e5)

  rasterToMatch <- prepInputsLCC(year = 2005, destinationPath = paths$inputPath,
                                 studyArea = studyArea) |>
    disagg(fact = 2)
  rasterToMatchLarge <- prepInputsLCC(year = 2005, destinationPath = paths$inputPath,
                                      studyArea = studyAreaLarge) |>
    disagg(fact = 2)

  studyArea <- st_transform(studyArea, crs = crs(rasterToMatch))
  studyAreaLarge <- st_transform(studyAreaLarge, crs = crs(rasterToMatch))
  studyAreaReporting <- st_transform(studyAreaReporting, crs = crs(rasterToMatch))
}

fireRegimePolys <- prepInputsFireRegimePolys(studyArea = studyArea,
                                             rasterToMatch = rasterToMatch,
                                             destinationPath = paths[["inputPath"]],
                                             type = "FRT")

fireRegimePolysLarge <- prepInputsFireRegimePolys(studyArea = studyAreaLarge,
                                                  rasterToMatch = rasterToMatchLarge,
                                                  destinationPath = paths[["inputPath"]],
                                                  type = "FRT")

objects <- list(
  studyArea = studyArea,
  studyAreaLarge = studyAreaLarge,
  studyAreaReporting = studyAreaReporting,
  rasterToMatch = rasterToMatch,
  rasterToMatchLarge = rasterToMatchLarge,
  fireRegimePolys = fireRegimePolys,
  fireRegimePolysLarge = fireRegimePolysLarge
)

outSim <- simInitAndSpades(times = list(start = 1, end = 5000),
                           params = parameters,
                           modules = modules,
                           objects = objects,
                           paths = paths)
```

Experiment

```
newLandscape <- randomStudyArea(size = 6.25*10000 * 1000 * 1000) ## this is 10k m2/ha * 6.25 ha/pixel *
newLandscape$PolyID <- 139
newLandscape <- st_as_sf(newLandscape)
newLandscapeLarge <- st_buffer(newLandscape, 5e3)
newLandscape <- st_as_sf(newLandscape)
newFlamAreaLarge <- raster(extent(newLandscapeLarge),
                           crs = crs(newLandscapeLarge),
                           res = c(250, 250))
newFlamAreaLarge[] <- rbinom(n = ncell(newFlamAreaLarge), size = 1, prob = 0.99)
newFlamAreaLarge <- mask(newFlamAreaLarge, newLandscapeLarge)
newFlamArea <- mask(newFlamAreaLarge, newLandscape)
newLandscapeAttr <- list("139" = outSim$landscapeAttr$`138`)
newLandscapeAttr$`139`$cellsByZone <- which(!is.na(newFlamArea[]))
newRegime <- list("139" = outSim$scfmRegimePars$`139`)
newFireRegimeRas <- newFlamArea
newFireRegimeRas[!is.na(newFireRegimeRas)] <- 139

newObjects <- list("fireRegimePolys" = newLandscape,
                   "flammableMapLarge" = newFlamAreaLarge,
                   "flammableMap" = newFlamArea,
                   "fireRegimeRas" = newFireRegimeRas,
                   "rasterToMatch" = newFlamArea,
                   "scfmRegimePars" = newRegime,
                   "landscapeAttr" = newLandscapeAttr)

outSim <- simInitAndSpades(times = times,
                           params = parameters,
                           modules = c("scfmDriver", "scfmIgnition", "scfmEscape", "scfmSpread"),
                           objects = newObjects,
                           paths = paths)
```

```
## TODO: add these plots as outputs in new scfmSummary module

dt <- scfmutils::comparePredictions_summaryDT(outSim)

## Some useful plots
gg_mfs <- scfmutils::comparePredictions_meanFireSize(dt)

# removed MAAB as diagnostic plot because it was derived from fire points incorrectly when SAL is suppl
# MAAB can still be calculated manually if a user desires

gg_fri <- scfmutils::comparePredictions_fireReturnInterval(dt)

gg_ign <- scfmutils::comparePredictions_annualIgnitions(dt)

gg_frp <- scfmutils::plot_fireRegimePolys(outSim$fireRegimePolys)

clearPlot()
gridExtra::grid.arrange(gg_frp,  gg_mfs,  gg_fri,  gg_ign, nrow = 2, ncol = 2)
```