scfm

Ian Eddy

December 18, 2018

Overview

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

Usage example

```
library(Require)

Require(c("data.table", "ggplot2",
          "PredictiveEcology/LandR@development",
          "magrittr", "raster", "SpaDES.core", "sf"))
```

```
## Using GITHUB_PAT to access files on GitHub
## Loading required package: data.table
## Loading required package: ggplot2
## Loading required package: LandR
## Loading required package: magrittr
## Loading required package: raster
## Loading required package: sp
## Loading required package: SpaDES.core
## Loading required package: quickPlot
## Loading required package: reproducible
##
## Attaching package: 'reproducible'
## The following object is masked from 'package:Require':
##
##     paddedFloatToChar
##
## Attaching package: 'SpaDES.core'
## The following object is masked from 'package:Require':
##
##     paddedFloatToChar
## The following objects are masked from 'package:stats':
##
##     end, start
```

```
## The following object is masked from 'package:utils':
##
##     citation
```

```
## Loading required package: sf
```

```
## Linking to GEOS 3.9.1, GDAL 3.3.2, PROJ 7.2.1; sf_use_s2() is TRUE
```

```
##              data.table (>= 1.11.0)                   ggplot2 (>= 3.4.0)
##                                TRUE                                 TRUE
## PredictiveEcology/LandR@development                  magrittr (>= 1.5)
##                                TRUE                                 TRUE
##                  raster (>= 3.5-21)                        SpaDES.core
##                                TRUE                                 TRUE
##                                  sf
##                                TRUE
```

```
# Parameters
timeunit <- "year"
times <- list(start = 1, end = 250)
defaultPlotInterval <- 50
defaultInitialSaveTime <- NA #don't be saving nuffink

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

parameters <- list(
  scfmLandcoverInit = list(
    useCache = globCache,
    sliverThreshold = 1e8), #polygons <100 km2 are merged with closest non-sliver neighbour
  scfmIgnition = list(
    .useCache = globCache
  ),
  scfmEscape = list(
    .useCache = globCache
  ),
  scfmSpread = list(
    .useCache = globCache,
    .plotInitialTime = times$start,
    .plotInterval = defaultPlotInterval),
  scfmRegime = list(
    .useCache = ".inputObjects",
    fireCause=c("L")
  ),
  scfmDriver = list(
    .useCache = '.inputObjects',
    targetN = 1000 #increase targetN for more robust estimates, longer run-time
  )# default targetN = 4000, for reference.
)


modules <- list(
  "scfmLandcoverInit", "scfmRegime", "scfmDriver",
  "scfmIgnition", "scfmEscape",
  # "scfmSpread", "scfmSummary"
)
## NOTE: replace modules with "group_scfm" to include ageModule;
```

```
##       ageModule isn't necessary to run and download of `ageMap` takes time

#Paths
paths <- list(
  cachePath = file.path("cache"),
  modulePath = file.path("modules"),
  inputPath = file.path("inputs"),
  outputPath = file.path("outputs")
)
#if you supply studyArea you should supply rtm to make sure the crs are identical.

center <- SpatialPoints(coords = data.frame(x = c(-1209980),
            y = c(7586895)), proj4string = CRS(paste("+proj=lcc +lat_1=49 +lat_2=77 +lat_0=0 +lon_0=-95
            "+datum=NAD83 +units=m +no_defs +ellps=GRS80 +towgs84=0,0,0")))
studyArea <- LandR::randomStudyArea(size = 10000 * 100 * 30000, center = center, seed = 1001)

studyAreaLarge <- buffer(studyArea, 50000)

rasterToMatchLarge <- raster(extent(studyAreaLarge), res = c(250, 250))
crs(rasterToMatchLarge) <- crs(studyAreaLarge)
rasterToMatchLarge[] <- 1
rasterToMatchLarge <- mask(rasterToMatchLarge, studyAreaLarge)
rasterToMatch <- postProcess(rasterToMatchLarge, studyArea = studyArea)

studyAreaLarge$name <- "SAL" #make SPDF

studyArea <- st_as_sf(studyArea)
studyAreaLarge <- st_as_sf(studyAreaLarge)
# rasterToMatchLarge <- terra::rast(rasterToMatchLarge)
# rastertoMatch <- terra::rast(rasterToMatch)
#if run with no studyArea, the default is a small area in southwest Alberta with very few fires
objects <- list(
  studyArea = studyArea,
  studyAreaLarge = studyAreaLarge,
  rasterToMatch = rasterToMatch,
  rasterToMatchLarge = rasterToMatchLarge
)

#Run module
#The calibration process may take hours (it's cached)
options("reproducible.cachePath" = paths$cachePath)
options("reproducible.useMemoise" = FALSE)

outSim <- simInitAndSpades(times = list(start = 1, end = 5),
                           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::compare Predictions_summaryDT(outSim)

## Some useful plots
gg_mfs <- scfmutils::compare Predictions_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::compare Predictions_fireReturnInterval(dt)

gg_ign <- scfmutils::compare Predictions_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)
```