Package: cure4insect (via r-universe)

August 27, 2024

Type Package

Title Custom Reporting for Intactness and Sector Effects

Version 0.2-2

Date 2020-11-03

Author Peter Solymos [cre, aut], Brandon Allen [aut], Ermias T. Azeria [aut], Shannon R. White [aut], ABMI [cph], BAM [cph]

Maintainer Peter Solymos <solymos@ualberta.ca>

Description Decision support tool that provides an interface to enable custom reporting for intactness and sector effects based on estimates and predictions created by the Alberta Biodiversity Monitoring Institute (ABMI) in collaboration with the Boreal Avian Modelling (BAM) Project.

URL https://github.com/ABbiodiversity/cure4insect

BugReports https://github.com/ABbiodiversity/cure4insect/issues

License MIT + file LICENSE

LazyLoad yes

LazyData true

Depends R (>= 3.5.0), sp, raster

Imports methods, Matrix, intrval, pbapply (>= 1.3-4), KernSmooth, base64enc, mefa4, sendmailR, parallel

Suggests knitr, rmarkdown, jsonlite, rgdal

VignetteBuilder knitr

Repository https://psolymos.r-universe.dev

RemoteUrl https://github.com/ABbiodiversity/cure4insect

RemoteRef HEAD

RemoteSha e08b19729a0eb169bd3529f2de877e9a004ec7e7

Contents

`custom_report` *Core Functions for Custom Reporting*

Description

These functions load pre-processed data and calculate intactness and sector effects for custom regions and sets of species.

Usage

```
load_common_data(path=NULL, version=NULL)
is_loaded()
clear_common_data()
subset_common_data(id=NULL, species="all")
clear_subset_data()

load_species_data(species, boot=NULL, path=NULL, version=NULL)
calculate_results(y, level=0.9)
rasterize_results(y)

report_all(boot=NULL, path=NULL, version=NULL, level=0.9, cores=NULL)
flatten(x, ...)
## S3 method for class 'c4iraw'
flatten(x, raw_boot=FALSE, limit=NULL, ...)

custom_report(id=NULL, species="all",
    path=NULL, version=NULL, address=NULL,
    boot=NULL, level=0.9, cores=NULL,
    raw_boot=FALSE, limit=NULL)
custom_predict(species, xy, veg, soil,
    path=NULL, version=NULL, ...)

set_options(...)
overlay_polygon(ply)
get_all_id(mregion="both", nr=NULL, nsr=NULL, luf=NULL)
get_all_species(taxon = "all", mregion="both", habitat, status)
get_id_locations()
get_id_table(mregion="both", nr=NULL, nsr=NULL, luf=NULL)
get_species_table(taxon = "all", mregion="both", habitat, status)
get_version_info()
```

```
get_all_qsid()
qs2km(qsid)
make_subset_map()
get_subset_info()
get_subset_id()
get_subset_species()

dowload_data(dir, species="all", version=NULL, ...)

load_spclim_data(species, path=NULL, version=NULL)
get_levels()
## S3 method for class 'c4ispclim'
predict(object, xy, veg, soil,
    method="simple", ...)
predict_mat(object, ...)
## S3 method for class 'c4ispclim'
predict_mat(object, xy, veg, soil,
    method="simple",  ...)
combine_veg_soil(xy, veg, soil, method="simple")
make_multispecies_map(type=c("richness", "intactness"),
    path=NULL, version=NULL, clip=TRUE, limit=NULL,
    area="ha", pair_adj=2)
```

Arguments

`path`	path to a local copy of results or `NULL` (default, the value of `getOption("cure4insect")$baseurl`).
`version`	version of the results or `NULL` `NULL` (default, the value of `getOption("cure4insect")$version`).
`id`	character, IDs of the 1km x 1km spatial pixel units to be used for the custom summaries. The Row_Col field defines the IDs and links the raster cells in the geodatabase (`http://science.abmi.ca/reports/2017/grids/Grid1km_working.gdb.zip`) or CSV (`http://science.abmi.ca/reports/2017/grids/Grid1km_working.csv.zip`; with latitude/longitude in NAD_1983_10TM_AEP_Forest projection `http://spatialreference.org/ref/epsg/3402/`). If `id` is a matrix-like object, values of the 1st column are taken. `id` can be of class 'SpatialPolygons' defined by the 'sp' package. The supplied values are turned into character internally. Use `get_all_id` to see all possible values.
`qsid`	character, quarter section (QS) IDs in the `"MER-RGE-TWP-SEC-QS"` format, e.g. `"4-12-1-2-SE"`. The CSV file `http://science.abmi.ca/reports/2017/grids/Grid1km_working.csv.zip` has the corresponding coordinates in NAD_1983_10TM_AEP_Forest projection `http://spatialreference.org/ref/epsg/3402/`).
`species`	A list of species defined by the field SpeciesID in the table `http://science.abmi.ca/reports/2017/data/species-info.csv`. If species is a matrix-like object, values of the 1st column are taken. The following values are also accepted to define groups of species: `"all"` (all groups/species), or one of the taxonomic (`"birds"`, `"lichens"`, `"mammals"`, `"mites"`, `"mosses"`, `"vplants"`) or other (`"upland"`, `"lowland"`, `"native"`, `"nonnative"`) groups. For a combination of filters, see `get_all_species`.

`boot`	logical or `NULL`, if confidence intervals for abundance and intactness are desired. Defaults to `getOption("cure4insect")$boot` when `NULL`.
`level`	numeric, level for confidence interval, defaults to 90%.
`cores`	integer, number of cores used in forking (used on Unix/Linux OS) or number of cluster workers (on Windows). Defaults to `getOption("cure4insect")$cores` when `NULL`.
`y`	and input object from call to `load_species_data`.
`x`	and input object from call to `calculate_results`.
`raw_boot`	logical, if raw bootstrap abundance results should be returned.
`limit`	numeric (0-1), species with mean abundance less than 100 * `limit` percent of the maximum abundance (current and reference combined) are flagged and should not be used when calculating multi-species metrics because the species is not effectively present in the region. Defaults to `getOption("cure4insect")$limit` when `NULL`.
`address`	character, optional email address to send finished results to in the format `"name@domain.org"`. The default (`NULL`) is not to send an email.
`object`	and input object from call to `load_spclim_data`.
`xy`	a 'SpatialPoints' object defined by the 'sp' package with geographic coordinates corresponding to `veg` and `soil` (i.e. centroids of polygons).
`veg`, `soil`	factor, vegetation/soil classes. One of the two or both must be provided. Can be `NULL` or missing.
`method`	character, the method argument for raster value `extract`ion. `"simple"`: values for the cell a point falls in, `"bilinear"`: interpolated from the values of the four nearest raster cells.
`taxon`	character, return `"all"` species, or a subset (one or more of `"birds"`, `"lichens"`, `"mammals"`, `"mites"`, `"mosses"`, `"vplants"`).
`habitat`	character, one of `"upland"` or `"lowland"` to indicate expert based habitat associations in Alberta.
`status`	character, one of `"native"` or `"nonnative"` to indicate expert based status in Alberta.
`mregion`	character, modeling region (`"both"`, `"north"`, or `"south"`).
`nr`, `nsr`, `luf`	character vector, narural regions (nr), natural subregions (nsr), and land use framework regions (luf) of Alberta.
`ply`	an object of class 'SpatialPolygons' defined by the 'sp' package.
`clip`	logical, if the multi-species map needs to be clipped to the region bounds defined by the spatial subset.
`type`	character, the type of multi-species map to produce. `"intactness"` is the raster cell level average intactness across species, `"richness"` is probability of >0 observation in a 1 ha plot summed across species (expected species richness).
`area`, `pair_adj`	spatial scale (`"ha"` or `"km"`) and pair adjustment factor to turn bird densities into probability of observing non-zero count before calculating richness (sum of these probabilities).

`dir`	Character, a directory name to download the data to.
`...`	Arguments in `tag = value` form, or a list of tagged values. The tags are configuration settings as described below. For the `dowload_data` function, these arguments are passed to `download.file`

Details

Configuration is stored in the file `system.file("config/defaults.conf", package="cure4insect")`. Current options are: `path` (path to results), `version` (version of results), `verbose` (default is 1, value 0 suppresses the messages), `cores` (number of cores to use in parallel calculations, uses forking on Linux/Unix/Mac and socket clusters on Windows, default is 1), `limit` (abundance threshold for regional intactness calculations), `boot` (bootstrap based uncertainties used or not), `trunc` (quantile for truncating rasterized maps to avoid outliers), `sender` (email of sender), `subject` (subject of the email), and `body` (body of the email).

The multicore processing performance (not on Windows, where shared memory forked processes are not available) might be limited by memory and network speed/bandwidth. Local copy of the data is the surest way to boost performance.

Value

`load_common_data` loads common data to memory of the R session. `is_loaded` check if common data has been loaded previously.

`subset_common_data` subsets the common data and makes the information available in the R session. `make_subset_map` makes a raster map of the spatial selection, `get_subset_info` counts the number of species and 1 square km spatial pixel units in the selection. `get_subset_id` and `get_subset_species` returns the species and spatial IDs, respectively, based on the selection.

`load_species_data` load data of a single species and returns it.

`clear_common_data`, and `clear_subset_data` clears the respective environments.

`calculate_results` calculates intactness and sector effects results based on spatial subset definitions and single species data. The output is a list of class 'c4iraw'. The `flatten` method arranges the results from `calculate_results` into a 1-liner data frame.

`report_all` calculate results based on sequential calls to `load_species_data` for all species defined in the subset. Returns a list (class 'c4ilist') output from `calculate_results`.

`custom_report` wrapper function to load common data, subset spatial units and species list, calculate results for all species, and optionally send results as attachment of a email. Returns an object of class 'c4idf' (inheriting from data frame), rows representing species (flattened results, the Comment field flagging possible issues).

`overlay_polygon` selects spatial IDs based on a 'SpatialPolygons' object.

`rasterize_results` takes the single species data without any spatial subset. The output is a raster `stack` object with the following layers: NC (current abundance), NR (reference abundance), SI (intactness), SI2 (two-sided intactness), SE, and CV (bootstrap based standard error and coefficient of variation estimates for current abundance).

`make_multispecies_map` calculated multi-species maps of richness (1-ha based probability of occurrence summed across species) or intactness (mean pixel level intactness of species whose mean abundance in the subset region is at least 100 * `limit` percent of the provincial maximum abundance).

set_options sets the options and return previous values invisibly. dowload_data downloads the data to the hard drive so that it can be used later. Set the path in the file system.file("config/defaults.conf", package="cure4insect") to have permanent effect, or in each session using set_options.

get_all_id gets all possible spatial IDs. get_all_species gets "all" possible species IDs, or a subset of those IDs for a selected taxon. get_id_locations gets the 'SpatialPoints' object with geographic coordinates of the spatial IDs, get_species_table gets the lookup table for species, or a group of species as defined by the taxon argument. get_all_qsid gets all possible quarter section (QS) IDs, qs2km finds the 1 square km units corresponding to the quarter section IDs. QS IDs are composed as "MER-RGE-TWP-SEC-QS" (Meridian [4-6], Range[1-30], Township[1-127], Section[1-36], and Quarter Section [NE, NW, SE, SW]), e.g. "4-12-1-2-SE". All these require to run load_common_data() first.

load_spclim_data is used to load raster data for spatially explicit predictions by predict.c4ispclim. predict.c4ispclim returns a data frame with columns 'veg', 'soil', and 'comb' (combines 'veg' and 'soil' using combine_veg_soil as a weighted average). Note: for some species, either the 'veg' or 'soil' based estimates are unavailable. predict.c4ispclim returns NA for these and the combined results will be NA as well.

get_version_info returns detailed version info for each taxa, depending on the version set through options or load_common_data, including the following variables: taxon, version (year of production), 1st and last year of field sampling (yr_first, yr_last), field methodology, base year and version of human footprint inventory (hf), version of the backfilled vegetation layer (veg), the type of model used (model), and the number of species by taxon. load_common_data prints these info upon successfully loading common data objects.

get_levels returns a list with elements 'veg' for vegetation and 'soil' for soil class levels expected by predict.c4ispclim. The predict_clim method is similar, but expects veg and soil to be a composition matrix with rows corresponding to points in space (e.g. grid cell centroids) and columns to correspond to vegetation/soil classes, with cell values as areas or proportions (row standardized). The return value is a list with elements veg and soil, each containing either a matrix with same dimensions as the corresponding input, or NULL when the input is missing.

Author(s)

Peter Solymos <solymos@ualberta.ca>

See Also

plot_sector and plot_abundance for plots.

Examples

```
## Not run:
## workflow with 1 species --------------------
## ID is a vector of Row_Col IDs of 1km pixels
## species is a vector if species IDs
load_common_data()
is_loaded()
get_version_info() # important details about versions
## here is how to inspect all possible spatial and species IDs
str(get_all_id())
str(get_all_species())
```

```
plot(xy <- get_id_locations(), pch=".")
summary(xy)
str(get_species_table())
## define spatial and species IDs
Spp <- "Ovenbird"
ID <- c("182_362", "182_363", "182_364", "182_365", "182_366", "182_367",
    "182_368", "182_369", "182_370", "182_371", "182_372")
subset_common_data(id=ID, species=Spp)
y <- load_species_data("Ovenbird")
x <- calculate_results(y)
x
flatten(x)

## using quarter sections
Spp <- "Ovenbird"
QSID <- c("4-12-1-2-SE", "4-12-1-2-SW", "4-12-1-3-SE", "4-12-1-3-SW")
qs2km(QSID) # corresponding Row_Col IDs
subset_common_data(id=QSID, species=Spp)
y <- load_species_data(Spp)
flatten(calculate_results(y))

## using pre-defined planning/management regions
#ID <- get_all_id(nr=c("Boreal", "Foothills"))
ID <- get_all_id(luf="North Saskatchewan")
subset_common_data(id=ID)
plot(make_subset_map())

## workflow with multiple species ----------------
load_common_data() # use as before
## id and species can be defined using text files
Spp <- read.table(system.file("extdata/species.txt", package="cure4insect"))
ID <- read.table(system.file("extdata/pixels.txt", package="cure4insect"))
subset_common_data(id=ID, species=Spp)
xx <- report_all()
str(xx)
do.call(rbind, lapply(xx, flatten))

## ID can also be a SpatialPolygons object based on GeoJSON for example
#library(rgdal)
#dsn <- system.file("extdata/polygon.geojson", package="cure4insect")
#ply <- readOGR(dsn=dsn)
#subset_common_data(id=ply, species=Spp)
#xx2 <- report_all()

## wrapper function ----------------------
## species="all" runs all species
## species="mites" runs all mite species
## sender="you@example.org" will send an email with the results attached
## increase cores to allow parallel processing
z <- custom_report(id=ID,
    species=c("AlderFlycatcher", "Achillea.millefolium"),
    address=NULL, cores=1)
z
```

```
## making of the file raw_all.rda
opar <- set_options(path = "w:/reports")
getOption("cure4insect")
load_common_data()
SPP <- get_all_species()
subset_common_data(id=get_all_id(), species=SPP)
res <- list()
for (i in 1:length(SPP)) {
    cat("processing species:", SPP[i], i, "/", length(SPP), "\n")
    flush.console()
    y <- load_species_data(SPP[i])
    res[[i]] <- calculate_results(y)
}
names(res) <- SPP

## spatial maps
y <- load_species_data("Ovenbird")
r <- rasterize_results(y)
plot(r, "NC") # current abundance map
plot(r, "SE") # standadr errors for current abundance

## making multi-species richness and intactness maps for birds
subset_common_data(species=get_all_species(taxon="birds"))
r1 <- make_multispecies_map("richness")
r2 <- make_multispecies_map("intactness")

## End(Not run)

## working with a local copy of the results is much faster
## set path via function arguments or the options:
getOption("cure4insect")
(opar <- set_options())
set_options(path = "/your/path/to/local/copy")
(set_options(opar)) # reset options

## change configs in this file to make it permanent for a given installation
as.list(drop(read.dcf(file=system.file("config/defaults.conf",
package="cure4insect"))))

## Not run:
## spatially explicit prediction
load_common_data()
## see bird species codes
sptab <- get_species_table()
rownames(sptab)[sptab$taxon == "birds"]
## pick Ovenbird
species <- "Ovenbird"
object <- load_spclim_data(species)
## vegetation/disturbance classes: use as factor
## might need to make a crosswalk, use e.g. mefa4::reclass
(veg <- as.factor(get_levels()$veg))
## for each veg class value, need to have
## spatial locations (can repeat the same value,
## but avoid duplicate rownames)
## use the sp package to get SpatialPoints as here:
XY <- get_id_locations()
coords <- coordinates(XY)[10^5,,drop=FALSE]
rownames(coords) <- NULL
xy <- data.frame(coords[rep(1, length(veg)),])
coordinates(xy) <- ~ POINT_X + POINT_Y
proj4string(xy) <- proj4string(XY)
## predict
pred <- predict(object, xy=xy, veg=veg)
summary(pred)

## End(Not run)
```

internals *App Specific Functions and Internals*

Description

These functions are needed by OpenCPU apps in the package, or needed to be exported by the package but not intended to be used by users.

Usage

```
app_read_csv(...)
app_test(...)

.load_species_data(species, boot=NULL, path=NULL, version=NULL,
    taxon, model_north, model_south)
.calculate_results(y, level=0.9, .c4is)
.calculate_limit(y, limit=NULL)
.report_all_by1(boot=NULL, path=NULL, version=NULL, level=0.9, cores=NULL)
.read_raster_template()
.make_raster(value, rc, rt)
.rasterize_multi(y, type=c("richness", "intactness"), rt)
.verbose()
.get_cores(cores=NULL)
.check(x, ref)
.combine_veg_soil(w, veg, soil)
.validate_id(id, type=c("km", "qs"))
.verbose()
.select_id(mregion="both", nr=NULL, nsr=NULL, luf=NULL)
p_bird(D, area=c("ha", "km"), pair_adj=2)
.truncate(x, trunc=NULL)
```

Arguments

`species`, `boot`, `path`, `version`	arguments passed to `load_species_data` and other functions.
`taxon`, `model_north`, `model_south`	taxonomic group required for the correct path, and logical values indicating north and south model results.
`y`, `level`	arguments passed to `calculate_results` and other functions.
`.c4is`	the subset environment `as.list(.c4is)`.
`cores`	desired number of cores to use.
`value`, `rc`, `rt`	call `value`, `rc` is data frame with `Row` and `Col` indices for `value`, `rt` is raster template from `.read_raster_template`.
`id`	spatial IDs.
`type`	type of multi-species map (richness or intactness) or spatial ID (km or QS).
`limit`	abundance threshold for multi-species intactness reporting.
`x`	input object.
`ref`	checking the validity of land cover classes in `x` against the `reference` list of possible values.
`w`, `veg`, `soil`	calculates a weighted average of `veg` and `soil` based models based on weights for the vegetation models (`w`).
`mregion`, `nr`, `nsr`, `luf`	regions.
`D`, `area`, `pair_adj`	density, scale, and pair adjustment to turn bird densities into probability of observing non-zero count.
`trunc`	quantile for truncating values of `x`, usually a vector or a raster object.
`...`	arguments passed to underlying functions.

Details

`app_read_csv` wraps `read.csv`.

`app_test` mimics `custom_report`.

Interface for internal functions might change and usage is not recommended.

Author(s)

Peter Solymos <solymos@ualberta.ca>

plot_sector *Abundance and Sector Effects Plots*

Description

Plots the sector effects for a single or a group of species.

Usage

```
plot_sector(x, ...)
## S3 method for class 'c4iraw'
plot_sector(x,
    type=c("unit", "regional", "underhf"), main, ylab, subset=NULL, ...)
## S3 method for class 'c4idf'
plot_sector(x,
    type=c("unit", "regional", "underhf"), main, ylab, subset=NULL, ...)

plot_intactness(x, ...)
## S3 method for class 'c4idf'
plot_intactness(x, type=c("SI", "SI2"), col, ...)

plot_abundance(species, type, plot=TRUE, paspen=0, ...)
```

Arguments

`x`	an object of class 'c4iraw' (from `calculate_results`) or a data frame (class 'c4idf') with flattened (1-row) results per species.
`species`	character, species ID (see `get_all_species`).
`type`	type of the plot, see Details.
`main`	title for the plot, if single species results are displayed the default is to use the species ID.
`ylab`	character, optional label for the y axis.
`subset`	subset of sectors to be plotted, can be any suitable index.
`plot`	logical, if a plot is to be drawn.
`paspen`	numeric in [0, 1], the probability of aspen occurrence (proxy for climatic suitability for treed vegetation).
`col`	color.
`...`	other possible arguments passed to underlying functions, e.g. `ylim`, `xlab`, `ylab`, `col` (for sector colors), or `method` (one of `"kde"`, `"fft"`, or `"hist"`).

Details

"unit" type sector effects are based on regional current and reference abundances, and the regional sector effects are standardized by footprint area.

"regional" sector effects includes native and disturbed habitats when comparing regional abundance under current and reference conditions.

"underhf" (under human footprint) type sector effects consider only the abundance that us 'under the footprint', meaning that the current designation is disturbed.

The single species sector effect plots are different kinds of bar plots. The multi-species plot represents violin (carrot, vase) plots based on kernel density, fast Fourier transform, or binning (histogram).

Intactness plots are either one sided (0-100%, "SI"), or two-sided (0-200%, "SI2") differentiating increased (>100%) and decreaser (<100%) species.

Abundance plots depend on the type argument: "veg_coef" type abundance plots show relative abundances across various land cover (incl. disturbance) classes, "soil_coef" shows relative abundance by soil and disturbance types. The "veg_lin" and "soil_lin" types show average relative abundances compared to 10% vegetated (soft) and non-vegetated (hard) linear disturbance.

Value

Called for the side effect of drawing a plot. Returns the plotted data invisibly.

Author(s)

Peter Solymos <solymos@ualberta.ca>

See Also

calculate_results

Examples

```
## Not run:
## *res*ults from calculate_results, all province, all species
fn <- paste0("http://science.abmi.ca/reports/",
    getOption("cure4insect")$version, "/misc/raw_all.rda")
con <- url(fn)
load(con)
close(con)

plot_sector(res[["CanadaWarbler"]], "unit")
plot_sector(res[["CanadaWarbler"]], "regional")
plot_sector(res[["CanadaWarbler"]], "underhf")

z <- do.call(rbind, lapply(res, flatten))
class(z) <- c("c4idf", class(z))
plot_sector(z, "unit") # all species
plot_sector(z[1:100,], "regional") # use a subset
plot_sector(z, "underhf", method="hist") # binned version
```

```
plot_intactness(z, "SI")
plot_intactness(z, "SI2", method="hist")

## land cover associations
load_common_data()
plot_abundance("Achillea.millefolium", "veg_coef")
plot_abundance("Achillea.millefolium", "soil_coef")
plot_abundance("Achillea.millefolium", "veg_lin")
plot_abundance("Achillea.millefolium", "soil_lin")

## R markdown file with worked examples
file.show(system.file("doc/example-species-report.Rmd", package="cure4insect"))

## End(Not run)
```

Index