Package 'ediblecity'

July 16, 2023

Type Package

Title Modeling Urban Agriculture at City Scale

Version 0.2.1

Description The purpose of this package is to estimate the potential of urban agriculture to contribute to addressing several urban challenges at the city-scale. Within this aim, we selected 8 indicators directly related to one or several urban challenges. Also, a function is provided to compute new scenarios of urban agriculture. Methods are described by Pueyo-Ros, Comas & Corominas (2023) <doi:10.12688/openreseurope.16054.1>.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

RoxygenNote 7.2.3

Imports sf (>= 0.9), dplyr (>= 1.0.6), magrittr (>= 2.0.1), stars (>= 0.5), rlang (>= 1.0)

Depends R (>= 2.10)

Suggests knitr, rmarkdown, testthat (>= 3.0.0)

Config/testthat/edition 3

VignetteBuilder knitr

URL https://github.com/icra/ediblecity, https://icra.github.io/ediblecity/

BugReports https://github.com/icra/ediblecity/issues

NeedsCompilation no

Author Josep Pueyo-Ros [aut, cre] (<https://orcid.org/0000-0002-1236-5651>), ICRA - Catalan Institute for Water Research [fnd] (Edicitnet, 776665)

Maintainer Josep Pueyo-Ros <josep.pueyo@udg.edu>

Repository CRAN

Date/Publication 2023-07-16 17:20:10 UTC

R topics documented:

`city_example` *City example*

Description

It contains a urban model as an example to use the package. It is specifically the urban model for Sant Narcís, a neighbourhood of Girona (Catalonia)

Author(s)

Josep Pueyo-Ros

`city_land_uses` *City Land Uses*

Description

It contains information about the functions belonging to green infrastructure and urban agriculture in the city_example dataset. This information can be used as params in the indicators, in some cases these are already the default values.

Author(s)

Josep Pueyo-Ros

edible_jobs *The jobs created by urban agriculture in your city*

Description

This indicator estimates the number of full-time jobs created by commercial urban agriculture initiatives in your city. It uses a range of jobs per square meter to create the median and the confidence interval of the number of jobs by simulating a random uniform distribution of 1000 values within the provided range.

Usage

```
edible_jobs(
  x,
  jobs = c(0.000163, 0.022),
  edible = NULL,
  area_col = "edible_area",
  interval = 0.95,
  verbose = FALSE
)
```

Arguments

x	An 'sf' object with the urban model of your city and a 'land_use' column with categories of urban features.
jobs	A vector of length 2 with the range of jobs created by square meter of edible gardens.
edible	The categories in 'land_uses' that represent commercial edible gardens. If NULL, the land_uses from 'city_land_uses' dataset are used where jobs is TRUE.
area_col	The column to be used as the area of each feature. If NULL, the area is calculated with sf::st_area()
interval	A numeric value with the confidence interval returned by the function.
verbose	If TRUE, the indicators returns a vector (N=1000) with all simulated values.

Value

If verbose is FALSE, it returns a named vector with the median and the low and high confidence intervals. Otherwise, it returns a vector of length 1000 with all simulated values.

Author(s)

Josep Pueyo-Ros

Examples

```
# First, we set a scenario with commercial gardens that create jobs
scenario <- set_scenario(city_example, pCommercial = 1, quiet = TRUE)
# Get the 95% confidence interval
edible_jobs(scenario, interval = 0.95)

# Get the raw values from the Monte Carlo simulation and adjust the number of jobs by square meter.
result <- edible_jobs(scenario, jobs = c(0.02, 0.03), verbose = TRUE)
result[1:10]
```

---

edible_volunteers *The number of volunteers involved in urban agriculture in your city*

Description

This indicator estimates the number of volunteers potentially involved in community urban agriculture initiatives in your city. It uses a range of volunteers per square meter to create the median and the confidence interval of the number of volunteers by simulating a random uniform distribution of 1000 values within the provided range. The default range came from required work hours in urban agriculture assessed in scientific literature, assuming that a volunteers dedicates a 10

Usage

```
edible_volunteers(
  x,
  volunteers = c(0.00163, 0.22),
  edible = NULL,
  area_col = "edible_area",
  interval = 0.95,
  verbose = FALSE
)
```

Arguments

x	An 'sf' object with the urban model of your city and a 'land_use' column with categories of urban features.
volunteers	A vector of length 2 with the range of volunteers involved by square meter of edible gardens.
edible	The categories in 'land_uses' that represent community edible gardens. If NULL, land_uses from 'city_land_uses' dataset area used where volunteers is TRUE.
area_col	The column to be used as the area of each feature. If NULL, the area is calculated with sf::st_area().
interval	A numeric value with the confidence interval returned by the function.
verbose	If TRUE, the indicators returns a vector (N=1000) with all simulated values.

Value

If verbose is FALSE, it returns a named vector with the median and the low and high confidence intervals. Otherwise, it returns a vector of length 1000 with all simulated values.

Author(s)

Josep Pueyo-Ros

Examples

```
# Get the 95% confidence interval
edible_volunteers(city_example, interval = 0.95)

# Get the raw values from the Monte Carlo simulation
# and adjust the number of volunteers by squared meter.
result <- edible_volunteers(city_example, volunteers = c(0.1, 0.2), verbose = TRUE)
result[1:10]
```

---

food_production *The food produced by urban agriculture in your city*

Description

This indicator estimates the food (in kg/year) produced by urban agriculture initiatives in your city. It uses a range of production for each type of initiative to create the median and the confidence interval of the number of jobs by simulating a random uniform distribution of 1000 values within the provided range.

Usage

```
food_production(
  x,
  edible_df = NULL,
  area_col = "edible_area",
  interval = 0.95,
  verbose = FALSE
)
```

Arguments

x	An 'sf' object with the urban model of your city and a 'land_use' column with categories of urban features.
edible_df	A dataframe of categories that are considered as urban agriculture with three columns: 1. 'land_uses': Column with the land_use to be considered in the calculations corresponding to 'land_use' column in 'x'. 2. 'food1': The low range of food production in each land_use (in kg/year/m2).

3. 'food2': The high range of food production of each land_use (in kg/year/m2).

area_col	The column to be used as the area of each feature. If NULL, the area is calculated with sf::st_area()
interval	A numeric value with the confidence interval returned by the land_use.
verbose	If TRUE, the indicators returns a vector (N=1000) with all simulated values.

Value

If verbose is FALSE, it returns a named vector with the median and the low and high confidence intervals (in kg/year). Otherwise, it returns a vector of length 1000 with all simulated values (in kg/year)

Author(s)

Josep Pueyo-Ros

Examples

```
# Estimate the food production within 95% confidence interval
food_production(city_example, interval = 0.95, verbose = FALSE)

# Get the raw values instead of the confidence interval
result <- food_production(city_example, verbose = TRUE)
result[1:10]
```

---

green_capita *Urban green per capita*

Description

This indicators calculates the amount of green per capita in the city. This may include private green such as gardens and crops or exclude them.

Usage

```
green_capita(
  x,
  green_categories = NULL,
  inhabitants = NULL,
  neighbourhoods = NULL,
  inh_col = NULL,
  name_col = NULL,
  private = FALSE,
  verbose = FALSE,
  min_inh = 0
)
```

Arguments

x	An 'sf' object with the urban model of your city and a 'land_use' column with categories of urban features.
green_categories	The categories that are considered as urban green. If NULL, categories of 'city_land_uses' are considered.
inhabitants	A value representing the inhabitants in the city.
neighbourhoods	(optional) An 'sf' object with polygons representing the neighborhoods in the city.
inh_col	(optional) The col in 'x' or in 'neighborhoods' indicating the inhabitants in each neighborhood.
name_col	(optional) The col in 'x' or in 'neighborhoods' indicating the name of each neighborhood
private	If FALSE (default), only public areas are considered in the indicator. If TRUE, elements in 'city_land_uses' where 'private' is TRUE are considered. Alternatively, a vector with land_uses to be considered.#'
verbose	If FALSE (default), the indicator returns the proportion between the most and the least green neighbourhoods. Otherwise, it will return a tibble with the green per capita in each neighborhood, provided that 'inh_col' and 'name_col' are provided.
min_inh	If neighbourhoods are used, those with less inhabitants than 'min-inh' will be discarded.

Details

If 'inh_col' and 'name_col' are defined and 'neighbourhoods' is NULL, the function searches the columns in 'x'. If 'neighbourhoods' is defined along with previous both, the columns are searched in 'neighbourhoods' and spatially joined with 'x'. In both cases, 'inhabitants' is ignored.

Value

A numeric value expressing the square meters of green per capita. Or a numeric value expressing the proportion between the greenest and the least green neighbourhood. Or a tibble with the green area, inhabitants and green per capita in each neighbourhood.

Author(s)

Josep Pueyo-Ros

Examples

```
# Calculate total green per capita in the city
green_capita(city_example, inhabitants = 6000)

# Calculate the differences between the greenest and the least green neighbourhoods
green_capita(city_example, neighbourhoods = neighbourhoods_example,
             inh_col = "inhabitants", name_col = "name")
```

```
# Get the green per capita in each neighbourhood
green_capita(city_example, neighbourhoods = neighbourhoods_example,
             inh_col = "inhabitants", name_col = "name", verbose = TRUE)

# Use a customized vector of land_uses to be considered private green
green_capita(city_example, inhabitants = 6000, private = c("Normal garden", "Commercial garden"))
```

`green_distance` *Distance to closest public green area*

Description

This indicator calculates the distance from each residence to its closest public green area larger than 'min_area'. It can return the summary of distances or the percentage of residence buildings further than a defined distance.

Usage

```
green_distance(
  x,
  green_cat = NULL,
  min_area = 5000,
  residence_col = "land_use_verbose",
  residences = "Residence",
  percent_out = FALSE,
  max_dist = 300,
  verbose = FALSE
)
```

Arguments

`x`	An 'sf' object with the urban model of your city and a 'land_use' column with categories of urban features.
`green_cat`	A vector with the categories in 'land_use' that must be considered in the calculations. If NULL (default), the 'city_land_uses' dataset is used where 'public' is TRUE.
`min_area`	A numerical value (in meters). smaller green areas are not considered in the calculations.
`residence_col`	The column 'x' where the residences are specified.
`residences`	A vector with the categories that represent residences.
`percent_out`	If TRUE, the function returns the percentage of residences further than 'max_dist'. (default = FALSE)
`max_dist`	A numeric value representing the maximum distance that a residence should be from a public green area.
`verbose`	If TRUE returns the vector of distances. Otherwise, it returns as specified in value section.

Value

If 'percent_out' is FALSE, it returns a summary of statistics for distance. Otherwise, it returns a numeric value with the percentage of residences further than 'max_dist' from its closest public green area.

Author(s)

Josep Pueyo-Ros

Examples

```
# Calculate a summary of the distances to closest public green area larger than 0.5 ha.
green_distance(city_example, min_area = 5000)

# Get the distances from each residence to its closest public green area.
result <- green_distance(city_example, min_area = 0, verbose = TRUE)
result[1:10]

# Get the percentage of residences further than 300 m. from a green area larger than 0.5 ha.
green_distance(city_example, percent_out = TRUE, max_dist = 300)
```

---

neighbourhoods_example *Neighbourhoods of City Example*

Description

It contains a GIS layer with the neighbourhoods of city_example and their population. It is used to run 'green_capita' indicator.

Author(s)

Josep Pueyo-Ros

---

no2_seq *Sequestration of nitrogen dioxide*

Description

This indicator returns the amount of NO2 that is sequestered by urban green.

Usage

```
no2_seq(x, green_df = NULL)
```

Arguments

x	An 'sf' object with the urban model of your city and a 'land_use' column with categories of urban features.
green_df	A dataframe of categories that are considered as urban green with four columns: 1. 'land_uses': Column with the function to be considered in the calculations corresponding to 'land_use' column in 'x'. 2. 'no2_seq1': The low range of NO2 sequestration of each function (in ug/s/m2). 3. 'no2_seq2': The high range of NO2 sequestration of each function (in ug/s/m2). 4. 'pGreen': The proportion of green surface in each function (0:1). This is overriden by 'edible_are' when land_uses are community garden, commercial garden, rooftop garden and hydroponic rooftop. If NULL, the 'city_land_uses' dataset is used.

Value

A numeric value with the total NO2 sequestration in the city (in grams/second).

Author(s)

Josep Pueyo-Ros

Examples

```
# Get the total nitrogen dioxide sequestered by urban green
no2_seq(city_example)
```

---

runoff_prev *Runoff prevention*

Description

The indicator calculates the runoff prevention considering a rain event, the infiltration capacity and the rain harvesting and storage capacity. government.

Usage

```
runoff_prev(
  x,
  runoff_df = NULL,
  rain = 85,
  floors_field = "floors",
  harvest_dist = 10,
  tank_size = c(0, 45)
)
```

Arguments

x	An 'sf' object with the urban model of your city and a 'land_use' column with categories of urban features.
runoff_df	A dataframe of categories that are considered impervious area with three columns. 1. 'land_uses' with the names of 'land_use' in 'x' to be considered as impervious. 2. Curve numbers columns 'CN1' and 'CN2' the range of curve number of that function. 3. 'water_storage', a boolean column indicating whether the land_uses is potentially harvesting and storing rainwater using a tank. If NULL, categories and values of 'city_land_uses' are considered.
rain	The amount of 24h-rain to be simulated, default is 85 mm.
floors_field	The column in 'x' containing the number of floors of each building. Zero is considered unbuilt areas like gardens or streets. It is used to calculate rainwater harvesting area, since only upper surface are considered. Missing values are considered as zero.
harvest_dist	Maximum distance (in meters) of buildings where to harvest rainwater
tank_size	A two-length vector with the range of tank size possibilities, proportional to the surface of each element where the tank size is located (in l/m2).

Value

It returns a named vector with values of runoff in mm, total rainfall and harvested rainwater in cubic meters.

Author(s)

Josep Pueyo-Ros

Examples

```
# Get the total values of runoff, rainfall and rain harvested
runoff_prev(city_example)

# Adjust the parameters for rain, maximum distance to harvest rainwater and tank size
runoff_prev(city_example, rain = 160, harvest_dist = 5, tank_size = c(20,30))
```

---

set_scenario *Set the scenario for your edible city*

Description

You can adjust different parameters to define different city scenarios. The object must contain a field 'land_use' which describes the function or type of each feature.

Usage

```
set_scenario(
  x,
  pGardens = 1,
  pVacant = 1,
  pRooftop = 1,
  edible_area_garden = c(0.02, 0.3),
  edible_area_vacant = c(0.52, 0.75),
  edible_area_rooftop = c(0.6, 0.62),
  min_area_garden = 10,
  min_area_vacant = 100,
  min_area_rooftop = 100,
  private_gardens_from = "Normal garden",
  vacant_from = "Vacant",
  rooftop_from = "Rooftop",
  pCommercial = 0,
  area_field = "flat_area",
  quiet = FALSE
)
```

Arguments

x	An 'sf' object with the urban model of your city and a 'land_use' field with categories of urban features.
pGardens	The proportion of private gardens (land_use == 'Gardens') that will become edible gardens [0-1].
pVacant	The proportion of vacant plot (land_use == 'Vacant') with 'area >= min_area_vacant' that will become edible gardens [0-1].
pRooftop	The proportion of rooftops (land_use == 'Flat rooftop') with 'area >= min_area_rooftop' that will become edible rooftops [0-1].
edible_area_garden	The proportion in a range of surface in a garden that is occupied by edible plants [0-1].
edible_area_vacant	The proportion in a range of surface in a vacant plot that is occupied by edible plants [0-1].
edible_area_rooftop	The proportion in a range of surface in a rooftop that is occupied by edible plants [0-1].
min_area_garden	The minimum area that a garden must have to become an edible garden.
min_area_vacant	The minimum area that a vacant must have to become an community or commercial garden.
min_area_rooftop	The minimum area that a flat rooftop must have to become an edible rooftop.

`private_gardens_from`
: The categories in 'land_uses' potentially converted to edible private gardens

`vacant_from`
: The categories in 'land_uses' potentially converted to community or commercial gardens

`rooftop_from`
: The categories in 'land_uses' potentially converted to edible rooftop (community raised beds or commercial hydroponic)

`pCommercial`
: The proportion of plots and rooftop that will be commercial. The rest will be community gardens In rooftops it is equivalent to raised beds and hydroponic system respectively.

`area_field`
: The field to be used as the area of each feature. If NULL, the area is calculated with sf::st_area()

`quiet`
: If 'TRUE', warnings about proportions not satisfied are not triggered.

Details

When pGardens, pVacant or pRooftop is lower than 1, the gardens are selected randomly among gardens with an area larger than 'min_area_*'. However, when pCommercial > 0, commercial gardens and hydroponic rooftops are settled in the larger features, assuming that commercial initiatives have the power to acquire the best spots.

Value

An 'sf' object as 'x' with the respective proportion of gardens ('Edible private garden'), vacant plots ('Community plot garden', 'Commercial plot garden') and rooftop gardens ('Community rooftop garden', 'Commercial hydroponic rooftop') labeled as edible gardens.

Author(s)

Josep Pueyo-Ros

Examples

```
# Set scenario with 50% of streets converted to community gardens
# randomly occupying between 40 and 60% of street's area.
scenario <- set_scenario(city_example, pGardens = 0, pVacant = 0.5, pRooftop = 0,
                         edible_area_vacant = c(0.4, 0.6), vacant_from = "Streets")
table(scenario$land_use)

# Set scenario with 60% of rooftops converted to gardens, and 30% of those with commercial purpose.
scenario <- set_scenario(city_example, pGardens = 0, pVacant = 0, pRooftop = 0.6, pCommercial = 0.3)
table(scenario$land_use)
```

SVF *Sky View Factor for City Example*

Description

It contains a raster representing the sky view factor for 'city_example', calculated using SAGA algorithm on a digital surface model of 3x3 from a LIDAR dataset.

Author(s)

Josep Pueyo-Ros

UHI *Heat island effect*

Description

The indicator calculates the urban heat island (UHI) using the DPRA guidelines of the Dutch government.

Usage

```
UHI(
  x,
  SVF,
  green_df = NULL,
  Qql = 6.11,
  Cair = 1007,
  Pair = 1.14,
  Tmax = 30.8,
  Tmin = 20,
  windspeed = 2.77,
  return_raster = FALSE,
  verbose = FALSE
)
```

Arguments

x	An 'sf' object with the urban model of your city and a 'land_use' column with categories of urban features.
SVF	A 'stars' object representing sky view factor. It can be computed, e.g. with SAGA's Sky View Factor algorithm and then loaded with stars::read_stars().
green_df	A dataframe of categories that are considered as urban green with two columns. 'land_uses' with the names of 'land_use' in 'x' to be considered as green; a 'pGreen' column with the percentage of green of that function. If NULL, categories and values of 'city_land_uses' dataset are considered.

Qgl	A numerical value representing the average solar radiation in W/m2/hour.
Cair	A numerical value representing the air heat capacity in J.
Pair	A numerical value representing the air density in kg/m3.
Tmax	Averaged maximum temperature in ºC.
Tmin	Averaged minimum temperature in ºC.
windspeed	Averaged wind speed in m/s.
return_raster	If TRUE, the raster of UHI values is returned. Otherwise, a summary of raster values is returned.
verbose	If TRUE, returns a vector with UHI value in each cell.

Details

DEFAULT values are the values for 'city_example' dataset in August (averaged values from 2011-2020)

Value

A 'stars' object with values of UHI. Or a numerical vector or summary statistic for UHI values. See params for more information on how to select each one.

Author(s)

Josep Pueyo-Ros

Examples

```
# Get a summary of the UHI
UHI(city_example, SVF)

# Get a 'stars' object representing UHI
uhi <- UHI(city_example, SVF, return_raster = TRUE)
plot(uhi)
```

Index