Package 'ClimMobTools'

September 12, 2019

Type Package

Title Tools for Crowdsourcing Citizen Science in Agriculture

Version 0.2-7

Maintainer Kaue de Sousa <kaue.desousa@inn.no>

URL https://agrobioinfoservices.github.io/ClimMobTools/

BugReports https://github.com/agrobioinfoservices/ClimMobTools/issues

Description Toolkit for the 'ClimMob' platform in R. 'ClimMob' is an open source software for crowdsourcing citizen science in agriculture <https://climmob.net/climmob3/>. Developed by van Etten et al. (2019) <doi:10.1017/S0014479716000739>, it turns the research paradigm on its head; instead of a few researchers designing complicated trials to compare several technologies in search of the best solutions, it enables many farmers to carry out reasonably simple experiments that taken together can offer even more information. 'ClimMobTools' enables project managers to deep explore and analyse their 'ClimMob' data in R.

License GPL-3

Encoding UTF-8

LazyData true

Depends R (>= 2.10)

Imports httr, jsonlite, Matrix, methods, nasapower, PlackettLuce (>= 0.2-8), raster, RSpectra, tibble, tidyr, utils

Suggests psychotree, psychotools, qvcalc, knitr, rmarkdown, testthat (>= 2.1.0)

Language en-GB

RoxygenNote 6.1.1

VignetteBuilder knitr

NeedsCompilation no

Author Kaue de Sousa [aut, cre] (<https://orcid.org/0000-0002-7571-7845>), Jacob van Etten [aut, ctb] (<https://orcid.org/0000-0001-7554-2558>), Sam Dumble [aut, ctb], Nicolas Greliche [aut, ctb],

Brandon Madriz [aut, ctb],
Carlos Quiros [aut, ctb]

Repository CRAN

Date/Publication 2019-09-12 15:40:05 UTC

R topics documented:

`build_rankings` *Build Plackett-Luce rankings*

Description

Create an object of class "rankings" from ClimMob data

Usage

```
build_rankings(data = NULL, items = NULL, input = NULL,
  additional.rank = NULL, group = FALSE, ...)
```

Arguments

`data`	a data frame with columns specified by items and input values to rank.
`items`	a data frame or index of `data` for the column(s) containing the item names
`input`	a data frame or index of `data` for the column(s) containing the values to be ranked
`additional.rank`	optional, a data frame for the comparisons between tricot items and the local item
`group`	logical, if TRUE return an object of class "grouped_rankings"
`...`	additional arguments passed to methods. See details

Details

full.output: logical, to return a list with a "rankings", a "grouped_rankings" and the ordered items

Value

a PlackettLuce "rankings" object, which is a matrix of dense rankings

See Also

rankings

Examples

```
# beans data where each observer compares 3 varieties randomly distributed
# from a list of 11 and additionally compares these 3 varieties
# with their local variety
library("PlackettLuce")
data("beans", package = "PlackettLuce")

# first build rankings with only tricot items
# and return an object of class 'rankings'
R <- build_rankings(data = beans,
                    items = c(1:3),
                    input = c(4:5))
head(R)

############################################################

# pass the comparison with local item as an additional rankings, then
# each of the 3 varieties are compared separately with the local item
# and return an object of class grouped_rankings
G <- build_rankings(data = beans,
                    items = c(1:3),
                    input = c(4:5),
                    additional.rank = beans[c(6:8)],
                    group = TRUE)

head(G)

############################################################

# rankings with five items in a ClimMob project

items <- randomise(5, 5, 5, c("green","blue","red","white","yellow"))

input <- as.data.frame(matrix(NA, nrow = 5, ncol = 5))
names(input) <- paste0("position_item_",LETTERS[1:5])

for(s in 1:5) {
  input[s,] <- sample(1:5)
}

build_rankings(items = items,
               input = input)
```

ETo *Evapotranspiration*

Description

Compute evapotranspiration using the Blaney-Criddle method

Usage

```
ETo(object, day.one = NULL, span = NULL, lat = NULL, Kc = 1,
  p = NULL)
```

Arguments

object	a numeric vector of geographic coordinates (lonlat) or an array with two dimensions containing the temperature data; 1st dimension contains the day temperature and 2nd dimension the night temperature. When lonlat is used, the function makes a call to `nasapower::get_power` to fetch and concatenate environmental data from NASA POWER (https://power.larc.nasa.gov/)
day.one	a vector of class 'Date' for the starting date to capture the environmental data
span	an integer or a vector with integers for the duration of the timespan to be captured
lat	a vector for the latitude (in Decimal degrees)
Kc	a numeric value for the crop factor for water requirement
p	optional, a numeric value (from 0 to 1) used if lat is not given, representing the mean daily percentage of annual daytime hours for different latitudes

Value

The evapotranspiration in mm/day

Examples

```
library("ClimMobTools")
library("nasapower")

# random geographic locations around bbox(11, 12, 55, 58)
lonlat <- data.frame(lon = runif(10, 11, 12),
                     lat = runif(10, 55, 58))

# random planting dates around 2018-05-15 and 2018-05-20
pdates <- as.integer(runif(10, 17666, 17670))
pdates <- as.Date(pdates, origin = "1970-01-01")
```

```
# the evapotranspiration in the first 50 days after planting
ETo(lonlat,
    day.one = pdates,
    span = 50,
    lat = lonlat[["lat"]])
```

GDD *Growing Degree Days*

Description

Compute number of days required to reach growing degree days. GDD are calculated by taking the integral of warmth above a base temperature.

Usage

```
GDD(object, day.one = NULL, degree.days = NULL, base = NULL,
  span = NULL, ...)
```

Arguments

object	a numeric vector of geographic coordinates (lonlat) or an array with two dimensions containing the temperature data; 1st dimension contains the day temperature and 2nd dimension the night temperature. When lonlat is used, the function makes a call to `nasapower::get_power` to fetch and concatenate environmental data from NASA POWER (https://power.larc.nasa.gov/)
day.one	a vector of class 'Date' for the starting date to capture the environmental data
degree.days	an integer for the degree days required by the crop (look at the physiology of the focal crop)
base	an integer for the base temperature. Set 10 as default.
span	an integer or a vector with integers for the duration of the timespan to be captured
...	additional arguments passed to methods

Value

The number of days required to reach the growing degree days.

Examples

```
library("ClimMobTools")
library("nasapower")

# random geographic locations around bbox(11, 12, 55, 58)
lonlat <- data.frame(lon = runif(10, 11, 12),
                     lat = runif(10, 55, 58))

# random planting dates around 2018-05-15 and 2018-05-20
pdates <- as.integer(runif(10, 17666, 17670))
pdates <- as.Date(pdates, origin = "1970-01-01")

# Calculate the days required for the plants in these plots to reach the
# maturity. The crop requires ~1800 degree days for it.

GDD(lonlat,
    day.one = pdates,
    degree.days = 1800,
    base = 5)
```

`getDataCM` *Get ClimMob data*

Description

Fetch the trial data from a ClimMob project using an API key

Usage

```
getDataCM(key = NULL, project = NULL, tidynames = FALSE,
  pivot.wider = FALSE, ...)
```

Arguments

`key`	a character for the ClimMob user API key
`project`	a character for the project id
`tidynames`	logical, if TRUE suppress ODK strings
`pivot.wider`	logical, if TRUE return a wider object, each observer is a row
`...`	additional arguments passed to methods

Value

A data frame with the project data

`id`	the participant's package id
`moment`	the data collection moment
`variable`	the variable name
`value`	the value for each variable

See Also

`getProjectsCM`

Examples

```
## Not run:

# This function will not work without an API key
# the user API key can be obtained once a free ClimMob account
# is created via https://climmob.net/climmob3/

my_key <- "add_your_key"
my_project <- "my_climmob_project"

data <- getDataCM(key = my_key, project = my_project)


## End(Not run)
```

`getProjectsCM` *Get ClimMob projects*

Description

Fetch the list of ClimMob projects using your API key

Usage

```
getProjectsCM(key = NULL)
```

Arguments

`key`	a character for the ClimMob user API key

Value

A data frame with the ClimMob projects

project_id	the project unique id
name	the project name
status	the current status
creation_date	the project's creation date

See Also

https://climmob.net/climmob3/

Examples

```
## Not run:
# This function will not work without an API key
# the user API key can be obtained once a free ClimMob account
# is created via https://climmob.net/climmob3/

my_key <- "add_your_key"

getProjectsCM(key = my_key)


## End(Not run)
```

---

rainfall *Rainfall indices*

Description

Compute rainfall indices over a timespan

Usage

```
rainfall(object, day.one = NULL, span = NULL, index = NULL, ...)
```

Arguments

object	a numeric vector of geographic coordinates (lonlat) or a matrix containing the precipitation data. When lonlat is used, the function makes a call to nasapower::get_power to fetch and concatenate environmental data from NASA POWER (https://power.larc.nasa.gov/)
day.one	a vector of class 'Date' for the starting date to capture the environmental data
span	an integer or a vector with integers for the duration of the timespan to be captured

index	optional, a character or a vector with characters for the indices to be computed. All indices are given by default
...	additional arguments passed to methods

Value

A dataframe with selected indices. Options are:

MLDS	maximum length of consecutive dry days (r < 1 mm)
MLWS	maximum length of consecutive wet days (r >= 1 mm)
R10mm	number of heavy precipitation days (10 >= r < 20 mm)
R20mm	number of very heavy precipitation days (r >= 20)
SDII	simple daily intensity index (mean of wet days / total rainfall)
Rx1day	maximum 1-day rainfall (mm)
Rx5day	maximum 5-day rainfall (mm)
Rtotal	total rainfall (mm) in wet days (R >= 1)

Examples

```
library("ClimMobTools")
library("nasapower")

# random geographic locations around bbox(11, 12, 55, 58)
lonlat <- data.frame(lon = runif(10, 11, 12),
                     lat = runif(10, 55, 58))

# random planting dates around 2018-05-15 and 2018-05-20
pdates <- as.integer(runif(10, 17666, 17670))
pdates <- as.Date(pdates, origin = "1970-01-01")

# calculate rainfall for the first 50 days after planting
rainfall(lonlat,
         day.one = pdates,
         span = 50)


# include the first 15 days before planting (residual precipitation)
rainfall(lonlat,
         day.one = pdates,
         span = 50,
         days.before = 15)
```

randomise *Randomised group of items*

Description

Set a randomised group of items for crowdsourcing citizen science. Generate designs for ranking of options. It is designed for tricot trials specifically (comparing 3 options), but it will also work with comparisons of any other number of options. The design strives for approximate A optimality, this means that it is robust to missing observations. It also strives for balance for positions of each option. Options are equally divided between first, second, third, etc. position. The strategy is to create a "pool" of combinations that does not repeat combinations and is A-optimal. Then this pool is ordered to make subsets of consecutive combinations also relatively balanced and A-optimal

Usage

```
randomise(ncomp = 3, nobservers = NULL, nitems = NULL,
  itemnames = NULL)
```

Arguments

ncomp	an integer for the number of items each observer compares
nobservers	an integer for the number of observers
nitems	an integer for the number of items tested in the project
itemnames	a character for the name of items tested in the project

Value

A dataframe with the randomised design

Examples

```
ni <- 3
no <- 10
nv <- 4
inames <- c("mango","banana","grape","apple")

randomise(ncomp = ni,
          nobservers = no,
          nitems = nv,
          itemnames = inames)
```

seed_need *Required seed amount in a tricot project*

Description

Calculate the required amount of seeds (or other technology) required for a triadic comparison of technologies (tricot) project.

Usage

```
seed_need(nobservers = 100, ncomp = 3, nitems = 10, nseeds = 0.15,
  unit = "kg")
```

Arguments

nobservers	an integer for the number of observers
ncomp	an integer for the number of items each observer compares
nitems	an integer for the number of items tested in the project
nseeds	an integer for the metric of seeds each bag receives
unit	optional, a character specifying the metric unit used

Value

a dataframe with required number of seeds

Examples

```
# allocate 0.2 kg of seeds per variety in a project with 500
# participants and 14 varieties
seed_need(nobservers = 500,
          ncomp = 3,
          nitems = 14,
          nseeds = 0.2)

# allocate 100 seedlings per variety in a project with 400
# participants, 8 varieties and 3 comparisons between varieties
seed_need(nobservers = 400,
          ncomp = 3,
          nitems = 9,
          nseeds = 100,
          unit = "unit")
```

`temperature` *Temperature indices*

Description

Compute temperature indices over a timespan

Usage

```
temperature(object, day.one = NULL, span = NULL, index = NULL, ...)
```

Arguments

`object`	a numeric vector of geographic coordinates (lonlat) or an array with two dimensions containing the temperature data; 1st dimension contains the day temperature and 2nd dimension the night temperature. When lonlat is used, the function makes a call to `nasapower::get_power` to fetch and concatenate environmental data from NASA POWER (https://power.larc.nasa.gov/)
`day.one`	a vector of class 'Date' for the starting date to capture the environmental data
`span`	an integer or a vector with integers for the duration of the timespan to be captured
`index`	optional, a character or a vector with characters for the indices to be computed. All indices are given by default
`...`	additional arguments passed to methods

Value

A dataframe with selected indices. Options are:

`maxDT`	maximun day temperature (degree Celsius)
`minDT`	minimum day temperature (degree Celsius)
`maxNT`	maximun night temperature (degree Celsius)
`minNT`	minimum night temperature (degree Celsius)
`DTR`	diurnal temperature range (mean difference between DT and NT (degree Celsius))
`SU`	summer days, number of days with maximum temperature > 30 (degree Celsius)
`TR`	tropical nights, number of nights with maximum temperature > 25 (degree Celsius)

Examples

```
library("ClimMobTools")
library("nasapower")

# random geographic locations around bbox(11, 12, 55, 58)
lonlat <- data.frame(lon = runif(10, 11, 12),
                     lat = runif(10, 55, 58))

# random planting dates around 2018-05-15 and 2018-05-20
pdates <- as.integer(runif(10, 17666, 17670))
pdates <- as.Date(pdates, origin = "1970-01-01")

# get temperature indices for the first 40 days
temperature(lonlat,
            day.one = pdates,
            span = 40)
```

Index