Package: ClimMobTools (via r-universe)

November 1, 2026

Type Package

Title API Client for the 'ClimMob' Platform

Version 1.9.0

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

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

Description API client for 'ClimMob', an open source software for decentralized large-N trials with the 'tricot' approach <https://climmob.net/>. 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 for the target environment, it enables many participants to carry out reasonably simple experiments that taken together can offer even more information. 'ClimMobTools' lets project managers retrieve, explore and analyse their 'ClimMob' data in R, design trials with sequentially balanced incomplete blocks, and protect the privacy of participants' locations before data are shared.

License MIT + file LICENSE

Encoding UTF-8

LazyData false

Depends R (>= 3.5.0)

Imports httr, jsonlite, stats, utils

Suggests climatrends, gosset, knitr, rmarkdown, sf, PlackettLuce, testthat (>= 3.0.0)

Language en-US

RoxygenNote 7.3.2

VignetteBuilder knitr

Config/testthat/edition 3

Config/pak/sysreqs libssl-dev

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

Date/Publication 2026-11-01 11:53:13 UTC

RemoteUrl https://github.com/agrdatasci/climmobtools

RemoteRef HEAD

RemoteSha 2248279c3614d06913afb0df3e9cf6f73349882f

Contents

ClimMobTools 2
exportBlockData 3
exportMeasuredTraits 4
exportTrialMetadata 4
exportTricotRanks 5
exportVariablesDescription 6
getDataCM 7
getProjectProgress 8
getProjectsCM 9
getSurvey 10
getTraitList 11
randomize 12
rankTricot 14
rmGeoIdentity 15

Index **17**

`ClimMobTools` *API Client for the 'ClimMob' platform in R*

Description

API client for 'ClimMob', an open source software for decentralized large-N trials with the 'tricot' approach https://climmob.net/. 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 for the target environment, it enables many participants to carry out reasonably simple experiments that taken together can offer even more information. 'ClimMobTools' lets project managers retrieve, explore and analyse their 'ClimMob' data in R, design trials with sequentially balanced incomplete blocks, and protect the privacy of participants' locations before data are shared.

Author(s)

Kauê de Sousa and Jacob van Etten and Brandon Madriz

See Also

Useful links:

- Development repository: https://github.com/agrdatasci/ClimMobTools
- Static documentation: https://agrdatasci.github.io/ClimMobTools/
- Report bugs: https://github.com/agrdatasci/ClimMobTools/issues
- The ClimMob paper: doi:10.1016/j.compag.2023.108539

`exportBlockData` *Export block-level (participant) data*

Description

Creates a wide-format data.frame of participant (block) data for external export. The output excludes ranking variables and personally identifiable information (PII), such as names, phone numbers, and location metadata. Optionally includes geolocation if available.

Usage

```
exportBlockData(x, ...)
```

Arguments

`x`	An object of class `CM_list` containing raw ClimMob trial data.
`...`	further arguments passed to methods

Value

A data.frame with one row per block (participant), excluding PII and ranking data.

See Also

`rmGeoIdentity`

Other export functions: `exportMeasuredTraits()`, `exportTrialMetadata()`, `exportTricotRanks()`, `exportVariablesDescription()`

exportMeasuredTraits *Extract and format measured traits*

Description

Extracts non-ranking, plot-level trait data from a ClimMob trial object. The function detects variables ending in "_a", "_b", or "_c", infers the trait and collection moment, and reshapes the data into long format for analysis.

Usage

```
exportMeasuredTraits(x)
```

Arguments

x	An object of class `CM_list` containing raw ClimMob trial data.

Details

If no measured traits are found in the data (i.e., no variables ending in "_a", "_b", or "_c"), the function returns an empty `CM_df` object with valid column names.

Value

A data frame of class `CM_df` with one row per genotype observation, including block ID, plot, genotype name, collection moment, trait name, value, and value type. Returns an empty `CM_df` if no measured traits are present.

See Also

Other export functions: `exportBlockData()`, `exportTrialMetadata()`, `exportTricotRanks()`, `exportVariablesDescription()`

exportTrialMetadata *Generate metadata for tricot trial export*

Description

Extracts and compiles key metadata from a ClimMob trial object to support standardized data documentation and publication, including fields such as trial identifiers, geographic bounding box, crop and taxon information, and participant statistics.

Usage

```
exportTrialMetadata(x)
```

Arguments

x	An object of class CM_list containing raw ClimMob trial data.

Value

A named list containing metadata fields required for tricot data export, including study (trial) name, description, dates, bounding box, participant counts, and institutional information.

See Also

Other export functions: exportBlockData(), exportMeasuredTraits(), exportTricotRanks(), exportVariablesDescription()

---

exportTricotRanks *Extract and format tricot rankings for data export*

Description

Processes ClimMob ranking data into a standardized long-format data frame suitable for export to external databases and analysis. For each trait, the function extracts rank positions per genotype and block based on participant responses. Compatible with Dataverse and other FAIR data publication systems.

Usage

```
exportTricotRanks(
  x,
  pattern = c("_pos", "_neg"),
  items = c("package_item_A", "package_item_B", "package_item_C"),
  nmin = 0.2,
  ...
)
```

Arguments

x	An object of class CM_list containing raw ClimMob trial data.
pattern	character, the tricot ranking pattern
items	a character or numerical vector for indexing the column(s) containing the item names in data
nmin	numeric, the minimum number of complete cases required
...	Additional arguments passed to getTraitList().

Value

A data frame in long format containing tricot rankings, with one row per genotype and trait, including the block identifier, plot label (A/B/C), genotype name, trait label, and assigned rank position.

See Also

Other export functions: exportBlockData(), exportMeasuredTraits(), exportTrialMetadata(), exportVariablesDescription()

---

exportVariablesDescription

Generate descriptions for trial variables

Description

Extracts metadata for variables found in tricot block data and ranking data, combining question descriptors from the registry and assessments. This function harmonizes variable names and assigns relevant descriptions, value types, and placeholders for ontology and controlled vocabularies.

Usage

```
exportVariablesDescription(x, tricot_ranks, measured_traits, block_data)
```

Arguments

x	An object of class CM_list containing raw ClimMob trial data.
tricot_ranks	A data frame of tricot rankings generated by exportTricotRanks().
measured_traits	A data frame of plot data generated by exportMeasuredTraits().
block_data	A data frame of block-level data generated by exportBlockData().

Value

A data frame describing all trial variables, including name, description, value type, and optional fields for ontology and controlled vocabularies.

See Also

Other export functions: exportBlockData(), exportMeasuredTraits(), exportTrialMetadata(), exportTricotRanks()

`getDataCM` *Get ClimMob data*

Description

Fetch the data from a ClimMob project using an application programming interface (API) key

Usage

```
getDataCM(
  key,
  project,
  userowner,
  as.data.frame = TRUE,
  as.text = FALSE,
  server = "climmob3",
  ...
)

## S3 method for class 'CM_list'
as.data.frame(x, ..., tidynames = TRUE, pivot.wider = TRUE)
```

Arguments

`key`	character, the user's API key
`project`	character, the project id
`userowner`	character, username of project's owner
`as.data.frame`	logical, to return a data frame, as.data.frame = FALSE returns a list
`as.text`	logical, to return a text file that can be parsed to json
`server`	character, to indicate from which server the data will be retrieved. See details
`...`	additional arguments passed to methods
`x`	an object of class `CM_list`
`tidynames`	logical, if `TRUE`, makes clean column names
`pivot.wider`	logical, if `TRUE` return a wider object where each tricot package is a row

Details

`server`: options are: "climmob" or "1000farms"

Value

An object of class 'CM_list' or a text file or a data.frame with class "CM_df" with the variables:

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

Author(s)

Kauê de Sousa

See Also

Other GET functions: `getProjectProgress()`, `getProjectsCM()`, `getSurvey()`

Examples

```
# This function only works with an API key
# the API key can be obtained from your ClimMob account

library("ClimMobTools")
my_key = "d39a3c66-5822-4930-a9d4-50e7da041e77"

getDataCM(key = my_key,
          project = "breadwheat",
          userowner = "gosset",
          server = "climmob3")

# get in the long format

getDataCM(key = my_key,
          project = "breadwheat",
          userowner = "gosset",
          server = "climmob3",
          pivot.wider = FALSE)
```

`getProjectProgress` *Get project progress*

Description

Fetch the progress of a ClimMob project

Usage

```
getProjectProgress(key, project, userowner, server = "climmob3")
```

Arguments

`key`	character, the user's API key
`project`	character, the project id
`userowner`	character, username of project's owner
`server`	character, to indicate from which server the data will be retrieved. See details

Details

server: the default server is "climmob" used for clients of https://climmob.net/climmob3/, other options are:

"1000farms" for clients of https://1000farms.climmob.net/

Value

A list with number of submissions per assessment and submissions per assessment per enumerator

Author(s)

Kauê de Sousa

See Also

Other GET functions: getDataCM(), getProjectsCM(), getSurvey()

Examples

```
# This function only works with an API key
# the API key can be obtained from your ClimMob account

library("ClimMobTools")
my_key = "ff05a174-28d0-4a40-ab5a-35dc486133a6"

getProjectProgress(key = my_key,
                   project = "gina2024",
                   userowner = "student",
                   server = "1000FARMS")
```

---

getProjectsCM *Get ClimMob projects*

Description

Fetch the status of ClimMob projects

Usage

```
getProjectsCM(key, server = "climmob3", ...)
```

Arguments

key	character, the user's API key
server	character, to indicate from which server the data will be retrieved. See details
...	additional arguments passed to methods. See details

Details

`server`: the default server is "climmob" used for clients of https://climmob.net/climmob3/, other options are:

"1000farms" for clients of https://1000farms.climmob.net/

Value

A data.frame with the variables:

`project_id`	the ClimMob single id in the server database
`project_code`	the project's code from the ClimMob user
`project_name`	the project's name
`user_owner`	the account name that owns the project
`country`	the country of project's implementation
`status`	the current status
`creation_date`	date when the project was created

Author(s)

Kauê de Sousa

See Also

Other GET functions: `getDataCM()`, `getProjectProgress()`, `getSurvey()`

Examples

```
# This function only works with an API key
# the API key can be obtained from your ClimMob account

my_key = "ff05a174-28d0-4a40-ab5a-35dc486133a6"

getProjectsCM(key = my_key, server = "1000FARMS")
```

`getSurvey` *Retrieve survey questions from a ClimMob project*

Description

Retrieves the list of survey questions associated with a ClimMob project using the ClimMob API. The function sends a request to the `getListOfQuestionsByProject` endpoint and returns the survey questions as a data frame.

This function is useful for extracting questionnaire structures used in ClimMob projects, which can then be used for documentation, translation, validation, or integration into analysis pipelines.

If the API request fails (e.g., invalid project, unsupported language, or server error), the function prints a message and returns `NULL` instead of terminating the R session.

Usage

```
getSurvey(key, userowner, project, language)
```

Arguments

key	character, the user's API key
userowner	character, username of project's owner
project	character, the project id
language	Character. Language code for the questionnaire

Details

The function queries the ClimMob API endpoint: https://1000farms.climmob.net/api/getListOfQuestionsByProjec

The API response is returned as JSON and parsed into an R data frame. If the server returns an error or invalid JSON, the function reports the issue and safely returns NULL.

Value

A data frame containing the survey questions associated with the specified project and language. Returns NULL if the request fails.

See Also

Other GET functions: getDataCM(), getProjectProgress(), getProjectsCM()

Examples

```
# This function only works with an API key
# the API key can be obtained from your ClimMob account

library("ClimMobTools")
my_key = "d39a3c66-5822-4930-a9d4-50e7da041e77"
```

---

getTraitList *Organize trait ranks in ClimMob data*

Description

This function helps in identifying the traits assessed in the tricot project and validates the data returning a list with logical vectors to support the transformation of tricot rankings into a PlackettLuce ranking

Usage

```
getTraitList(data, pattern, trait.labels = NULL, ...)
```

Arguments

`data`	data.frame, the ClimMob data
`pattern`	character, the tricot ranking pattern
`trait.labels`	an optional character with clean trait labels
`...`	additional arguments, not implemented yet

Value

a list with trait validation data

Examples

```
require("gosset")

data("breadwheat", package = "gosset")

getTraitList(breadwheat, c("_best", "_worst"))
```

`randomize` *Set an experimental incomplete block design*

Description

Generate a sequentially balanced incomplete block design for tricot trials. Every prefix of the design is kept close to balanced, so the design stays good when only the first packages are distributed or returned.

Usage

```
randomize(
  npackages,
  itemnames,
  ncomp = 3,
  availability = NULL,
  props = NULL,
  ridge = 0.001,
  polish = FALSE,
  schedule = TRUE,
  ...
)

randomise(...)
```

Arguments

`npackages`	an integer for the number of incomplete blocks (packages)
`itemnames`	a character vector with the unique names of the items
`ncomp`	an integer for the number of items in each block (default 3)
`availability`	optional, an integer vector with the maximum number of plots for each item in *itemnames*
`props`	optional, a numeric vector with the desired proportions for each item in *itemnames*
`ridge`	small ridge added to the Laplacian to handle a disconnected early design. Disconnected components are then strongly penalised
`polish`	logical, if `TRUE` run a final local search on the position balance of the complete design. This can reduce the position balance of prefixes
`schedule`	logical, if `TRUE` (the default) use the replication schedule as primary key. If `FALSE`, select on A-optimality alone (slower, similar quality)
`...`	arguments of the previous algorithm, such as `comp`, are accepted for compatibility and ignored with a warning

Details

At each step the function

1. keeps only blocks whose items all have remaining quota (hard availability);
2. prefers blocks with the most items behind their replication schedule;
3. breaks ties by the exact A-optimality gain of the prefix, evaluated for all tied candidates with a rank-k update of the Laplacian inverse;
4. breaks remaining ties at random.

Item order within blocks is then balanced sequentially across positions. Call `set.seed()` before the function to make the design reproducible.

Value

A data.frame with one row per package and the columns item_A, item_B, ...

Author(s)

Jacob van Etten

References

Bailey and Cameron (2004). Combinations of optimal designs. https://webspace.maths.qmul.ac.uk/l.h.soicher/designtheory.org/library/preprints/optimal.pdf

Examples

```
set.seed(1)
design = randomize(npackages = 20,
                   itemnames = c("apple", "banana", "grape", "mango", "orange"))
table(unlist(design))

# limited availability of some items
set.seed(1)
randomize(npackages = 20,
          itemnames = c("apple", "banana", "grape", "mango", "orange"),
          availability = c(5, 8, 50, 50, 50))
```

`rankTricot` *Build Plackett-Luce rankings from tricot dataset*

Description

Create an object of class "rankings" from tricot data

Usage

```
rankTricot(
  data,
  items,
  input,
  group = FALSE,
  validate.rankings = FALSE,
  additional.rank = NULL,
  ...
)
```

Arguments

`data`	a data.frame with columns specified by items and input values
`items`	a character or numerical vector for indexing the column(s) containing the item names in `data`
`input`	a character or numerical vector for indexing the column(s) containing the values in `data` to be ranked
`group`	logical, if `TRUE` return an object of class "grouped_rankings"
`validate.rankings`	logical, if `TRUE` implements a check on ranking consistency looking for possible ties, NA or letters other than A, B, C. These entries are set to 0
`additional.rank`	optional, a data frame for the comparisons between tricot items and the local item
`...`	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" or "grouped_rankings" object

Author(s)

Kauê de Sousa and Jacob van Etten, with ideas from Heather Turner

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
if (require("PlackettLuce")){
  data("beans", package = "PlackettLuce")

  # first build rankings with only tricot items
  # and return an object of class 'rankings'
  R = rankTricot(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 = rankTricot(data = beans,
                 items = c(1:3),
                 input = c(4:5),
                 group = TRUE,
                 additional.rank = beans[c(6:8)])

  head(G)
}
```

`rmGeoIdentity` *Remove geographical identity*

Description

Build a buffer around a set of geographical coordinates and take a random point around the buffer. The function is used to omit the precise location of tricot participants while keeping a close distance to the trial environment.

Usage

```
rmGeoIdentity(longlat, dist = 0.015, nQuadSegs = 2L, seed = NULL, ...)
```

Arguments

longlat	a data.frame or matrix with geographical coordinates long lat
dist	numeric, buffer distance for all *longlat*
nQuadSegs	integer, number of segments per quadrant
seed	integer, the seed for random number generation. Leave it NULL (the default) for data that will be shared. The random displacement does not depend on the original coordinates, so anyone who knows the seed can reproduce it and recover the exact locations. Use a seed only for tests or internal reproducibility, and never publish it
...	further arguments passed to sf methods

Value

A data frame with the random coordinates long lat within the buffer

Examples

```
library("sf")

xy = matrix(c(11.097799, 60.801090,
              11.161298, 60.804199,
              11.254428, 60.822457),
            nrow = 3, ncol = 2, byrow = TRUE)

rmGeoIdentity(xy)

#' the function also handles NAs

xy2 = matrix(c(11.097799, 60.801090,
               NA, NA,
               11.161298, 60.804199,
               11.254428, 60.822457,
               11.254428, NA),
             nrow = 5, ncol = 2, byrow = TRUE)

rmGeoIdentity(xy2)
```

Index

∗ **GET functions**
  getDataCM, 7
  getProjectProgress, 8
  getProjectsCM, 9
  getSurvey, 10
∗ **export functions**
  exportBlockData, 3
  exportMeasuredTraits, 4
  exportTrialMetadata, 4
  exportTricotRanks, 5
  exportVariablesDescription, 6

as.data.frame.CM_list *(*getDataCM*)*, 7

ClimMobTools, 2
ClimMobTools-package *(*ClimMobTools*)*, 2

exportBlockData, 3, *4–6*
exportMeasuredTraits, *3*, 4, *5, 6*
exportTrialMetadata, *3*, 4, *4*, *6*
exportTricotRanks, *3, 4*, 5, *5, 6*
exportVariablesDescription, *3–5*, 6, *6*

getDataCM, 7, *9–11*
getProjectProgress, 8, *8*, *10, 11*
getProjectsCM, *8*, 9, *9*, *11*
getSurvey, *8, 9*, 10, *10*
getTraitList, 11

randomise *(*randomize*)*, 12
randomize, 12
rankTricot, 14
rmGeoIdentity, *3*, 15

sf, *16*