Package: rppo (via r-universe)

July 24, 2024

Title Access the Global Plant Phenology Data Portal

URL `https://docs.ropensci.org/rppo/`, `https://github.com/ropensci/rppo/`

BugReports `https://github.com/ropensci/rppo/issues`

Version 2.0

Maintainer John Deck <`jdeck88@gmail.com`>

Description Search plant phenology data aggregated from several sources and available on the Global Plant Phenology Data Portal.

Depends R (>= 3.4.0)

License GPL-2

Encoding UTF-8

RoxygenNote 7.2.3

Imports jsonlite, readr, httr, data.table, reshape2

Suggests testthat, rmarkdown, markdown, knitr, covr

VignetteBuilder knitr

LazyData true

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

RemoteUrl https://github.com/ropensci/rppo

RemoteRef master

RemoteSha 97ee9a35fc2308ef7e1b5fab11ae34795d48cc32

Contents

`rppo-package` *Access the Global Plant Phenology Data Portal*

Description

Access data from the global plant phenology data portal (PPO data portal) and phenology terms from the Plant Phenology Ontology (PPO)

Details

rppo enables users to query the global plant phenology data portal (PPO data portal). The PPO data portal is an aggregation of phenology data from several different data sources. Currently it contains USA-NPN, NEON, and PEP725 data sources. The PPO data portal harvests data using the ppo-data-pipeline, with code available at `https://github.com/biocodellc/ppo-data-pipeline/`. All phenological terms in the data portal are aligned using the Plant Phenology Ontology (PPO), available at `https://github.com/PlantPhenoOntology/ppo`.

Three main functions are contained in the **rppo**: `ppo_terms` allows users to discover present and absent phenological stages, `ppo_data` enables users to query the PPO data portal and `ppo_traits` use the data fetched from the PPO data portal and return the traits data for each eventID. The **rppo** package source code is available at `https://github.com/ropensci/rppo/`.

Author(s)

Maintainer: John Deck `<jdeck88@gmail.com>`

Authors:

- Brian Stucky `<stuckyb@flmnh.ufl.edu>`
- Ramona Walls `<rwalls@cyverse.org>`
- Kjell Bolmgren `<Kjell.Bolmgren@slu.se>`
- Ellen Denny `<ellen@usanpn.org>`
- Dubois Salix `<salixdubois@gmail.com>`
- Ellen Denny `<ellen@usanpn.org>`
- Robert Guralnick `<rguralnick@flmnh.ufl.edu>`

See Also

Useful links:

- `https://docs.ropensci.org/rppo/`
- `https://github.com/ropensci/rppo/`
- Report bugs at `https://github.com/ropensci/rppo/issues`

`ppo_data` *Access Data From the Global Plant Phenology Data Portal*

Description

Access data from the global plant phenology data portal (PPO data portal)

Usage

```
ppo_data(
  scientificName = NULL,
  genus = NULL,
  specificEpithet = NULL,
  termID = NULL,
  fromYear = NULL,
  toYear = NULL,
  fromDay = NULL,
  toDay = NULL,
  bbox = NULL,
  source = NULL,
  subSource = NULL,
  status = NULL,
  mapped_traits = NULL,
  eventRemarks = NULL,
  limit = 100000L,
  timeLimit = 60,
  keepData = FALSE
)
```

Arguments

`scientificName`	(character) A plant species scientific name.
`genus`	(character) A plant genus name. See details.
`specificEpithet`	(character) A plant specific epithet
`termID`	(character) A single termID from the plant phenology ontology. See details.
`fromYear`	(integer) return data from the specified year
`toYear`	(integer) return data up to and including the specified year
`fromDay`	(integer) return data starting from the specified day
`toDay`	(integer) return data up to and including the specified day
`bbox`	(character) return data within a bounding box. Format is `lat,long,lat,long` and is structured as a string. Use this website: http://boundingbox.klokantech.com/ to quickly grab a bbox (set format on bottom left to csv and be sure to switch the order from long, lat, long, lat to lat, long, lat, long).

`source`	(character) return data from specified source. See details.
`subSource`	(character) return data from the specified sub-source. See details.
`status`	(character) Either "present" or "absent". Return data with the specified status.
`mapped_traits`	(character) return data from the specified traits. See details
`eventRemarks`	(character) return data from the specified eventRemarks
`limit`	(integer) limit returned data to a specified number of records
`timeLimit`	(integer) set the limit of the amount of time to wait for a response
`keepData`	(logical) whether to keep (TRUE) or delete (FALSE; default) the downloaded data (~/ppo_download/).

Details

The ppo_data function returns a list containing the following information: a readme file, citation information, a data frame with data, an integer with the number of records returned and a status code. The function is called with parameters that correspond to values contained in the data itself which act as a filter on the returned record set. For a list of available mapped_traits, termID, Source and subSource see the `ppo_filters` dataset. For mapped_traits and termID, the `ppo_get_terms` function will return a data.frame with present, absent or both terms and traits information. The `ppo_terms` will do the same but will use the API to get the lastest data. However, some of the traits/termID may not return any results from this function. See their documentation for more details.

Value

A list with the following elements:

- ‘data‘: A data frame containing data
- ‘readme‘: A string with information about the return package
- ‘citation‘: A string with citation information
- ‘number_possible‘: An integer with total possible results
- ‘status_code‘: An integer with status code returned from server

Examples

```
r1 <- ppo_data(genus = c("Quercus", "Pinus"), termID='obo:PPO_0002313',
limit=10, timeLimit = 4)
head(r1$data)

r2 <- ppo_data(fromDay = 1, toDay = 100, bbox="37,-120,38,-119", limit=10,
timeLimit = 4)
head(r2$data)
```

`ppo_filters` *List of filters to use with* `ppo_data`.

Description

A list of the sources, sub-sources, status, traits and genera available and their number of observations in the PPO data portal.

Usage

```
ppo_filters
```

Format

A list with 5 element:

- sourceA data set with 1 row and 2 variable:
  - sourcethe source name
  - nObsthe number of obeservations
- subSourceA data set with 1 row and 2 variable:
  - subSourcethe subSource name
  - nObsthe number of obeservations
- statusA data set with 1 row and 2 variable:
  - statusthe status name
  - nObsthe number of obeservations
- mapped_traitsA data set with 1 row and 2 variable:
  - termIDthe term ID
  - labelthe term label
  - definitionthe term definition from Ontobee
  - urithe link to the term page from Ontobee
  - nObsthe number of obeservations
  - statusif the term status is ’present’ or ’absent; useful to filter by status
  - ObjectProperty_termIDthe term for an object propriety in the definition; useful to filter for similar terms
  - ObjectProperty_termID1the term for a second object propriety in the definition; useful to filter for similar terms
  - ObjectProperty_termID2the term for a third object propriety in the definition; useful to filter for similar terms

  ...

Source

https://www.ontobee.org

ppo_get_terms *Access Terms From the Plant Phenology Ontology*

Description

Access present and absent terms from the Plant Phenology Ontology

Usage

```
ppo_get_terms(present = FALSE, absent = FALSE)
```

Arguments

`present`	(boolean) If TRUE then return all "present" phenological stages
`absent`	(boolean) IF TRUE then return all "absent" phenological stages.

Details

The ppo_terms function returns terms from the Plant Phenology Ontology (PPO). The termID or label column can be used to query the 'termID' or 'mapped_traits' to `ppo_data`. The label and description columns are useful in determining the trait to query on. The URI column contains a link to the term itself which is useful for determining superclass and subclass relationships for each term.

Value

A data frame with the columns: termID, label, description, and URI.

Examples

```
presentTerms <- ppo_terms(present = TRUE)
absentTerms <- ppo_terms(absent = TRUE)
allTerms <- ppo_terms(present = TRUE, absent = TRUE)
fruitTerms <- grep("fruit", ppo_filters$mapped_traits$label, value = TRUE)
ppo_data(genus = c( "Pinus", "Quercus"),
         specificEpithet = "palustris",
         mapped_traits = fruitTerms,
         fromYear = 2016, toYear = 2017)
```

`ppo_terms` *Access Terms From the Plant Phenology Ontology*

Description

Access present and absent terms from the Plant Phenology Ontology

Usage

```
ppo_terms(present = FALSE, absent = FALSE, timeLimit = 4)
```

Arguments

`present`	(boolean) If TRUE then return all "present" phenological stages
`absent`	(boolean) IF TRUE then return all "absent" phenological stages.
`timeLimit`	(integer) set the limit of the amount of time to wait for a response

Details

The ppo_terms function returns terms from the Plant Phenology Ontology (PPO). The function only accepts parameters for "present" or "absent" terms. The response populates a data frame with: termID, label, description, and URI. Use the termID values in submitting termID values to the `ppo_data` function. The label and description fields are extracted from the Plant Phenology Ontology and are useful in determining the proper term to query on. The URI field contains a link to the term itself which is useful for determining superclass and subclass relationships for each term. Some of these terms will not return any results when using `ppo_data`. To have only the ones that will return results, use `ppo_get_terms` or check rppo:::ppo_filters$mapped_traits which contains both present and absent terms. For more information on the PPO ontology itself, we suggest loading the PPO `https://github.com/PlantPhenoOntology/ppo` with protege `https://protege.stanford.edu/`

Value

data.frame

Examples

```
presentTerms <- ppo_terms(present = TRUE, timeLimit = 1)

absentTerms <- ppo_terms(absent = TRUE, timeLimit = 1)
```

ppo_traits — *Get the traits data from the [ppo_data()] event ids.*

Description

Get the traits data from the [ppo_data()] event ids.

Usage

```
ppo_traits(x, sorted = TRUE, flatten_traits = TRUE, flatten_all = FALSE)
```

Arguments

`x`	(object) As returned from link(rppo)[ppo_data]
`sorted`	(logical) Should the output traits be sorted by category. Default : TRUE
`flatten_traits`	(logical) Should the traits list be melted in a data.frame. Default : TRUE
`flatten_all`	(logical) Should the output list be melted in a data.frame. Default : FALSE

Value

If sorted is TRUE, a list for each event id containing a list with the following elements:

- ‘metadata‘: A data frame containing metadata
- ‘taxonomy‘: A data.frame containing the taxonomy
- ‘traits‘: if flatten_traits is TRUE, a melted data.frame, else a list containing the traits value

Else, a list of data.frames for each event id. IF flatten_all is TRUE, the list is flatten to a data.frame

Examples

```
r1 <- ppo_data(genus = "Quercus", termID='obo:PPO_0002313', limit=10, timeLimit = 4)
r1_traits <- ppo_traits(r1)
```

ppo_traits_flatten — *Turn the traits element or the entire list returned by [pp_traits()] into a data.frame.*

Description

Turn the traits element or the entire list returned by [pp_traits()] into a data.frame.

Usage

```
ppo_traits_flatten(x, flatten_all = FALSE)
```

Arguments

`x`	(object) A list or data.frame as returned from link(rppo)[ppo_traits]
`flatten_all`	(logical) Should the output list be melted in a data.frame. Default : FALSE

Value

A list for each event id containing a list with the following elements:

- ‘metadata‘: A data frame containing metadata
- ‘taxonomy‘: A data.frame containing the taxonomy
- ‘traits‘: A melted data.frame containing the traits value

IF flatten_all is TRUE, the list is flatten to a data.frame

Examples

```
r1 <- ppo_data(genus = "Quercus", termID = 'obo:PPO_0002313', limit = 10, timeLimit = 4)
r1_traits <- ppo_traits(r1, sorted = FALSE, flatten_traits = FALSE)
r1_traits <- ppo_traits_flatten(r1_traits, flatten_all = TRUE)
```

`ppo_traits_sort` *Title*

Description

Title

Usage

```
ppo_traits_sort(x, flatten_traits = TRUE, flatten_all = FALSE)
```

Arguments

`x`	(object) A list or data.frame as returned from link(rppo)[ppo_traits] with sorted set to FALSE.
`flatten_traits`	(logical) Should the traits list be melted in a data.frame. Default : TRUE
`flatten_all`	(logical) Should the output list be melted in a data.frame. Default : FALSE

Value

A list for each event id containing a list with the following elements:

- ‘metadata‘: A data frame containing metadata
- ‘taxonomy‘: A data.frame containing the taxonomy
- ‘traits‘: if flatten_traits is TRUE, a melted data.frame, else a list containing the traits value

IF flatten_all is TRUE, the list is flatten to a data.frame

Examples

```
r1 <- ppo_data(genus = "Quercus", termID='obo:PPO_0002313', limit=10, timeLimit = 4)
r1_traits <- ppo_traits(r1,  sorted = FALSE)
r1_traits <- ppo_traits_sort(r1_traits)
```

Index