Package 'rnpn'

March 25, 2025

Title Interface to the National 'Phenology' Network 'API'

Version 1.4.0

Description Programmatic interface to the Web Service methods provided by the National 'Phenology' Network (<https://usanpn.org/>), which includes data on various life history events that occur at specific times.

License MIT + file LICENSE

URL https://github.com/usa-npn/rnpn, http://usa-npn.github.io/rnpn/

BugReports https://github.com/usa-npn/rnpn/issues

Depends R (>= 3.5.0)

Imports dplyr, httr2 (>= 1.1.0), jsonlite, lifecycle, magrittr, rlang, tibble, tidyr, xml2

Suggests covr, ggplot2, knitr, markdown, RColorBrewer, rmarkdown, sf, terra, testthat (>= 3.0.0), vcr, withr

VignetteBuilder knitr

Config/testthat/edition 3

Encoding UTF-8

RoxygenNote 7.3.2

NeedsCompilation no

Author Jeff Switzer [aut, cre],
Scott Chamberlain [aut],
Lee Marsh [aut],
Kevin Wong [aut],
Eric R Scott [aut] (<https://orcid.org/0000-0002-7430-7879>),
David LeBauer [ctb]

Maintainer Jeff Switzer <jeff@usanpn.org>

Repository CRAN

Date/Publication 2025-03-25 21:00:05 UTC

Contents

rnpn-package . . . 2
npn_abundance_categories . . . 3
npn_datasets . . . 4
npn_download_geospatial . . . 4
npn_download_individual_phenometrics . . . 5
npn_download_magnitude_phenometrics . . . 9
npn_download_site_phenometrics . . . 12
npn_download_status_data . . . 16
npn_get_agdd_point_data . . . 19
npn_get_custom_agdd_raster . . . 20
npn_get_custom_agdd_time_series . . . 21
npn_get_layer_details . . . 23
npn_get_phenophases_for_taxon . . . 24
npn_get_point_data . . . 25
npn_groups . . . 26
npn_lookup_names . . . 27
npn_phenophases . . . 27
npn_phenophases_by_species . . . 28
npn_phenophase_definitions . . . 29
npn_phenophase_details . . . 29
npn_pheno_classes . . . 30
npn_set_env . . . 31
npn_species . . . 31
npn_species_types . . . 33
npn_stations . . . 33
npn_stations_by_location . . . 34
npn_stations_by_state . . . 35
npn_stations_with_spp . . . 35
rnpn-defunct . . . 36

Index **37**

`rnpn-package` *Interface to the National Phenology Network API*

Description

This package allows for easy access to the National Phenology Network's Data API. To learn more, take a look at the vignettes.

Author(s)

Maintainer: Jeff Switzer <jeff@usanpn.org>

Authors:

- Scott Chamberlain
- Lee Marsh
- Kevin Wong
- Eric R Scott (ORCID)

Other contributors:

- David LeBauer [contributor]

See Also

Useful links:

- https://github.com/usa-npn/rnpn
- http://usa-npn.github.io/rnpn/
- Report bugs at https://github.com/usa-npn/rnpn/issues

`npn_abundance_categories` *Get Abundance Categories*

Description

Gets data on all abundance/intensity categories and includes a data frame of applicable abundance/intensity values for each category

Usage

```
npn_abundance_categories(...)
```

Arguments

`...`	Currently unused.

Value

A data frame listing all abundance/intensity categories and their corresponding values.

Examples

```
## Not run:
ac <- npn_abundance_categories()

## End(Not run)
```

npn_datasets *Get Datasets*

Description

Returns a complete list of information about all datasets integrated into the NPN dataset. Data can then be pulled for individual datasets using their unique IDs.

Usage

```
npn_datasets(...)
```

Arguments

...	Currently unused.

Value

tibble of datasets and their IDs.

Examples

```
## Not run:
npn_datasets()

## End(Not run)
```

npn_download_geospatial *Download Geospatial Data*

Description

Function for directly downloading any arbitrary Geospatial layer data from the NPN Geospatial web services.

Usage

```
npn_download_geospatial(
  coverage_id,
  date,
  format = "geotiff",
  output_path = NULL
)
```

Arguments

coverage_id	The coverage id (machine name) of the layer for which to retrieve. Applicable values can be found via the npn_get_layer_details() function under the name column.
date	Specify the date param for the layer retrieved. This can be a calendar date formatted YYYY-mm-dd or it could be a string integer representing day of year. It can also be NULL in some cases. Which to use depends entirely on the layer being requested. More information available from the npn_get_layer_details() function.
format	The output format of the raster layer retrieved. Defaults to "GeoTIFF".
output_path	Optional value. When set, the raster will be piped to the file path specified. When left unset, this function will return a terra::SpatRaster object.

Details

Information about the layers can also be viewed at the getCapbilities page directly: https://geoserver.usanpn.org/geoserver/wms?request=GetCapabilities

Value

returns nothing when output_path is set, otherwise a terra::SpatRaster object meeting the coverage_id, date and format parameters specified.

Examples

```
## Not run:
ras <- npn_download_geospatial("si-x:30yr_avg_six_bloom", "255")

## End(Not run)
```

---

npn_download_individual_phenometrics

Download Individual Phenometrics

Description

This function allows for a parameterized search of all individual phenometrics records in the USA-NPN database, returning all records as per the search parameters in a data table. Data fetched from NPN services is returned as raw JSON before being channeled into a data table. Optionally results can be directed to an output file in which case raw JSON is converted to CSV and saved to file; in that case, data is also streamed to file which allows for more easily handling of the data if the search otherwise returns more data than can be handled at once in memory.

Usage

```
npn_download_individual_phenometrics(
  request_source,
  years,
  period_start = "01-01",
  period_end = "12-31",
  coords = NULL,
  individual_ids = NULL,
  species_ids = NULL,
  station_ids = NULL,
  species_types = NULL,
  network_ids = NULL,
  states = NULL,
  phenophase_ids = NULL,
  functional_types = NULL,
  additional_fields = NULL,
  climate_data = FALSE,
  ip_address = NULL,
  dataset_ids = NULL,
  genus_ids = NULL,
  family_ids = NULL,
  order_ids = NULL,
  class_ids = NULL,
  pheno_class_ids = NULL,
  email = NULL,
  download_path = NULL,
  six_leaf_layer = FALSE,
  six_bloom_layer = FALSE,
  agdd_layer = NULL,
  six_sub_model = NULL,
  additional_layers = NULL,
  wkt = NULL
)
```

Arguments

`request_source`	Required field, character Self-identify who is making requests to the data service.
`years`	Required field, character vector. Specify the years to include in the search, e.g. `c('2013','2014')`. You must specify at least one year.
`period_start, period_end`	Character vectors of the form "MM-DD". Used to determine the period over which phenophase status records are summarized. For example, to use a "water year" set `period_start = "10-01"` and `period_end = "09-30"`. If not provided, they will default to "01-01" and "12-31", respectively, to use the calendar year.
`coords`	Numeric vector, used to specify a bounding box as a search parameter, e.g. `c(lower_left_lat, lower_left_long, upper_right, lat,upper_right_long)`.

`individual_ids` Comma-separated string of unique IDs for individual plants/animal species by which to filter the data.

`species_ids` Integer vector of unique IDs for searching based on species, e.g. `c(3, 34, 35)`.

`station_ids` Integer vector of unique IDs for searching based on site location, e.g. `c(5, 9)`.

`species_types` Character vector of unique species type names for searching based on species types, e.g. `c("Deciduous", "Evergreen")`.

`network_ids` Integer vector of unique IDs for searching based on partner group/network, e.g. `c(500, 300)`.

`states` Character vector of US postal states to be used as search params, e.g. `c("AZ", "IL")`.

`phenophase_ids` Integer vector of unique IDs for searching based on phenophase, e.g. `c(323, 324)`.

`functional_types`
Character vector of unique functional type names, e.g. 'c("Birds").

`additional_fields`
Character vector of additional fields to be included in the search results, e.g. `c("Station_Name", "Plant_Nickname")`.

`climate_data` Boolean value indicating that all climate variables should be included in `additional_fields`.

`ip_address` Optional field, string. IP Address of user requesting data. Used for generating data reports.

`dataset_ids` Integer vector of unique IDs for searching based on dataset, e.g. NEON or GRSM `c(17,15)`.

`genus_ids` Integer vector of unique IDs for searching based on taxonomic family, e.g. `c(3, 34, 35)`. This parameter will take precedence if `species_ids` is also set.

`family_ids` Integer vector of unique IDs for searching based on taxonomic family, e.g. `c(3, 34, 35)`. This parameter will take precedence if `species_ids` is also set.

`order_ids` Integer vector of unique IDs for searching based on taxonomic order, e.g. `c(3, 34, 35)`. This parameter will take precedence if `species_ids` or `family_ids` are also set.

`class_ids` Integer vector of unique IDs for searching based on taxonomic class, e.g. `c(3, 34, 35)`. This parameter will take precedence if `species_ids`, `family_ids` or `order_ids` are also set.

`pheno_class_ids`
Integer vector of unique IDs for searching based on pheno class. Note that if both `pheno_class_id` and `phenophase_id` are provided in the same request, `phenophase_id` will be ignored.

`email` Optional field, string. Email of user requesting data.

`download_path` Character, optional file path to the file for which to output the results.

`six_leaf_layer` Boolean value when set to `TRUE` will attempt to resolve the date of the observation to a spring index, leafing value for the location at which the observations was taken.

six_bloom_layer
: Boolean value when set to TRUE will attempt to resolve the date of the observation to a spring index, bloom value for the location at which the observations was taken.

agdd_layer
: Numeric value, accepts 32 or 50. When set, the results will attempt to resolve the date of the observation to an AGDD value for the location; the 32 or 50 represents the base value of the AGDD value returned. All AGDD values are based on a January 1st start date of the year in which the observation was taken.

six_sub_model
: Affects the results of the six layers returned. Can be used to specify one of three submodels used to calculate the spring index values. Thus setting this field will change the results of six_leaf_layer and six_bloom_layer. Valid values include: 'lilac', 'zabelli' and 'arnoldred'. For more information see the NPN's Spring Index Maps documentation: https://www.usanpn.org/data/maps/spring.

additional_layers
: Data frame with first column named name and containing the names of the layer for which to retrieve data and the second column named param and containing string representations of the time/elevation subset parameter to use. This variable can be used to append additional geospatial layer data fields to the results, such that the date of observation in each row will resolve to a value from the specified layers, given the location of the observation.

wkt
: WKT geometry by which filter data. Specifying a valid WKT within the contiguous US will filter data based on the locations which fall within that WKT.

Details

This data type includes estimates of the dates of phenophase onsets and ends for individual plants and for animal species at a site during a user-defined time period. Each row represents a series of consecutive "yes" phenophase status records, beginning with the date of the first "yes" and ending with the date of the last "yes", submitted for a given phenophase on a given organism. Note that more than one consecutive series for an organism may be present within a single growing season or year.

Most search parameters are optional, however, users are encouraged to supply additional search parameters to get results that are easier to work with. request_source must be provided. This is a self-identifying string, telling the service who is asking for the data or from where the request is being made. It is recommended you provide your name or organization name. If the call to this function is acting as an intermediary for a client, then you may also optionally provide a user email and/or IP address for usage data reporting later.

Additional fields provides the ability to specify additional, non-critical fields to include in the search results. A complete list of additional fields can be found in the NPN service's companion documentation Metadata on all fields can be found in the following Excel sheet: https://www.usanpn.org/files/metadata/individual_phenometrics_datafield_descriptions.xlsx

Value

A tibble of all status records returned as per the search parameters. If download_path is specified, the file path is returned instead.

Examples

```
## Not run:
#Download all saguaro data for 2013 and 2014 using "water year" as the period
npn_download_individual_phenometrics(
  request_source = "Your Name or Org Here",
  years = c(2013, 2014),
  period_start = "10-01",
  period_end = "09-30",
  species_id = 210,
  download_path = "saguaro_data_2013_2014.csv"
)

## End(Not run)
```

`npn_download_magnitude_phenometrics`

Download Magnitude Phenometrics

Description

This function allows for a parameterized search of all magnitude phenometrics in the USA-NPN database, returning all records as per the search results in a data table. Data fetched from NPN services is returned as raw JSON before being channeled into a data table. Optionally results can be directed to an output file in which case raw JSON is saved to file; in that case, data is also streamed to file which allows for more easily handling of the data if the search otherwise returns more data than can be handled at once in memory.

Usage

```
npn_download_magnitude_phenometrics(
  request_source,
  years,
  period_frequency = "30",
  coords = NULL,
  species_ids = NULL,
  genus_ids = NULL,
  family_ids = NULL,
  order_ids = NULL,
  class_ids = NULL,
  pheno_class_ids = NULL,
  station_ids = NULL,
  species_types = NULL,
  network_ids = NULL,
  states = NULL,
  phenophase_ids = NULL,
  functional_types = NULL,
  additional_fields = NULL,
```

```
  climate_data = FALSE,
  ip_address = NULL,
  dataset_ids = NULL,
  email = NULL,
  download_path = NULL,
  taxonomy_aggregate = NULL,
  pheno_class_aggregate = NULL,
  wkt = NULL
)
```

Arguments

request_source	Required field, character Self-identify who is making requests to the data service.
years	Required field, character vector. Specify the years to include in the search, e.g. `c('2013','2014')`. You must specify at least one year.
period_frequency	Required field, integer. The integer value specifies the number of days by which to delineate the period of time specified by the start_date and end_date, i.e. a value of 7 will delineate the period of time weekly. Any remainder days are grouped into the final delineation. This parameter, while typically an int, also allows for a "special" string value, "months" to be passed in. Specifying this parameter as "months" will delineate the period of time by the calendar months regardless of how many days are in each month. Defaults to `30` if omitted.
coords	Numeric vector, used to specify a bounding box as a search parameter, e.g. `c(lower_left_lat, lower_left_long, upper_right, lat,upper_right_long)`.
species_ids	Integer vector of unique IDs for searching based on species, e.g. `c(3, 34, 35)`.
genus_ids	Integer vector of unique IDs for searching based on taxonomic family, e.g. `c(3, 34, 35)`. This parameter will take precedence if `species_ids` is also set.
family_ids	Integer vector of unique IDs for searching based on taxonomic family, e.g. `c(3, 34, 35)`. This parameter will take precedence if `species_ids` is also set.
order_ids	Integer vector of unique IDs for searching based on taxonomic order, e.g. `c(3, 34, 35)`. This parameter will take precedence if `species_ids` or `family_ids` are also set.
class_ids	Integer vector of unique IDs for searching based on taxonomic class, e.g. `c(3, 34, 35)`. This parameter will take precedence if `species_ids`, `family_ids` or `order_ids` are also set.
pheno_class_ids	Integer vector of unique IDs for searching based on pheno class. Note that if both `pheno_class_id` and `phenophase_id` are provided in the same request, `phenophase_id` will be ignored.
station_ids	Integer vector of unique IDs for searching based on site location, e.g. `c(5, 9)`.
species_types	Character vector of unique species type names for searching based on species types, e.g. `c("Deciduous", "Evergreen")`.
network_ids	Integer vector of unique IDs for searching based on partner group/network, e.g. `c(500, 300)`.

`states` Character vector of US postal states to be used as search params, e.g. `c("AZ", "IL")`.

`phenophase_ids` Integer vector of unique IDs for searching based on phenophase, e.g. `c(323, 324)`.

`functional_types`
Character vector of unique functional type names, e.g. 'c("Birds").

`additional_fields`
Character vector of additional fields to be included in the search results, e.g. `c("Station_Name", "Plant_Nickname")`.

`climate_data` Boolean value indicating that all climate variables should be included in `additional_fields`.

`ip_address` Optional field, string. IP Address of user requesting data. Used for generating data reports.

`dataset_ids` Integer vector of unique IDs for searching based on dataset, e.g. NEON or GRSM `c(17,15)`.

`email` Optional field, string. Email of user requesting data.

`download_path` Character, optional file path to the file for which to output the results.

`taxonomy_aggregate`
Boolean value indicating whether to aggregate data by a taxonomic order higher than species. This will be based on the values set in `family_ids`, `order_ids`, or `class_ids`. If one of those three fields are not set, then this value is ignored.

`pheno_class_aggregate`
Boolean value indicating whether to aggregate data by the pheno class ids as per the `pheno_class_ids` parameter. If the `pheno_class_ids` value is not set, then this parameter is ignored. This can be used in conjunction with `taxonomy_aggregate` and higher taxonomic level data filtering.

`wkt` WKT geometry by which filter data. Specifying a valid WKT within the contiguous US will filter data based on the locations which fall within that WKT.

Details

This data type includes various measures of the extent to which a phenophase for a plant or animal species is expressed across multiple individuals and sites over a user-selected set of time intervals. Each row provides up to eight calculated measures summarized weekly, bi-weekly, monthly or over a custom time interval. These measures include approaches to evaluate the shape of an annual activity curve, including the total number of "yes" records and the proportion of "yes" records relative to the total number of status records over the course of a calendar year for a region of interest. They also include several approaches for standardizing animal abundances by observer effort over time and space (e.g. mean active bird individuals per hour). See the Metadata window for more information.

Most search parameters are optional, however, failing to provide even a single search parameter will return all results in the database. Request_Source must be provided. This is a self-identifying string, telling the service who is asking for the data or from where the request is being made. It is recommended you provide your name or organization name. If the call to this function is acting as an intermediary for a client, then you may also optionally provide a user email and/or IP address for usage data reporting later.

Additional fields provides the ability to specify more, non-critical fields to include in the search results. A complete list of additional fields can be found in the NPN service's companion documentation. Metadata on all fields can be found in the following Excel sheet: https://www.usanpn.org/files/metadata/magnitude_phenometrics_datafield_descriptions.xlsx

Value

A tibble of the requested data. If a download_path was specified, the file path is returned.

Examples

```
## Not run:
#Download book all saguaro data for 2013
npn_download_magnitude_phenometrics(
  request_source="Your Name or Org Here",
  years=c(2013),
  species_id=c(210),
  download_path="saguaro_data_2013.csv"
)

## End(Not run)
```

---

npn_download_site_phenometrics

Download Site Phenometrics

Description

This function allows for a parameterized search of all site phenometrics records in the USA-NPN database, returning all records as per the search parameters in a data table. Data fetched from NPN services is returned as raw JSON before being channeled into a data table. Optionally results can be directed to an output file in which case raw JSON is converted to CSV and saved to file; in that case, data is also streamed to file which allows for more easily handling of the data if the search otherwise returns more data than can be handled at once in memory.

Usage

```
npn_download_site_phenometrics(
  request_source,
  years,
  period_start = "01-01",
  period_end = "12-31",
  num_days_quality_filter = "30",
  coords = NULL,
  species_ids = NULL,
  genus_ids = NULL,
  family_ids = NULL,
  order_ids = NULL,
```

```
  class_ids = NULL,
  pheno_class_ids = NULL,
  station_ids = NULL,
  species_types = NULL,
  network_ids = NULL,
  states = NULL,
  phenophase_ids = NULL,
  functional_types = NULL,
  additional_fields = NULL,
  climate_data = FALSE,
  ip_address = NULL,
  dataset_ids = NULL,
  email = NULL,
  download_path = NULL,
  six_leaf_layer = FALSE,
  six_bloom_layer = FALSE,
  agdd_layer = NULL,
  six_sub_model = NULL,
  additional_layers = NULL,
  taxonomy_aggregate = NULL,
  pheno_class_aggregate = NULL,
  wkt = NULL
)
```

Arguments

request_source	Required field, character Self-identify who is making requests to the data service.
years	Required field, character vector. Specify the years to include in the search, e.g. `c('2013','2014')`. You must specify at least one year.
period_start, period_end	Character vectors of the form "MM-DD". Used to determine the period over which phenophase status records are summarized. For example, to use a "water year" set `period_start = "10-01"` and `period_end = "09-30"`. If not provided, they will default to "01-01" and "12-31", respectively, to use the calendar year.
num_days_quality_filter	Required field, defaults to `30`. The integer value sets the upper limit on the number of days difference between the first Y value and the previous N value for each individual to be included in the data aggregation.
coords	Numeric vector, used to specify a bounding box as a search parameter, e.g. `c(lower_left_lat, lower_left_long, upper_right, lat,upper_right_long)`.
species_ids	Integer vector of unique IDs for searching based on species, e.g. `c(3, 34, 35)`.
genus_ids	Integer vector of unique IDs for searching based on taxonomic family, e.g. `c(3, 34, 35)`. This parameter will take precedence if `species_ids` is also set.
family_ids	Integer vector of unique IDs for searching based on taxonomic family, e.g. `c(3, 34, 35)`. This parameter will take precedence if `species_ids` is also set.

`order_ids` Integer vector of unique IDs for searching based on taxonomic order, e.g. `c(3, 34, 35)`. This parameter will take precedence if `species_ids` or `family_ids` are also set.

`class_ids` Integer vector of unique IDs for searching based on taxonomic class, e.g. `c(3, 34, 35)`. This parameter will take precedence if `species_ids`, `family_ids` or `order_ids` are also set.

`pheno_class_ids`
Integer vector of unique IDs for searching based on pheno class. Note that if both `pheno_class_id` and `phenophase_id` are provided in the same request, `phenophase_id` will be ignored.

`station_ids` Integer vector of unique IDs for searching based on site location, e.g. `c(5, 9)`.

`species_types` Character vector of unique species type names for searching based on species types, e.g. `c("Deciduous", "Evergreen")`.

`network_ids` Integer vector of unique IDs for searching based on partner group/network, e.g. `c(500, 300)`.

`states` Character vector of US postal states to be used as search params, e.g. `c("AZ", "IL")`.

`phenophase_ids` Integer vector of unique IDs for searching based on phenophase, e.g. `c(323, 324)`.

`functional_types`
Character vector of unique functional type names, e.g. 'c("Birds").

`additional_fields`
Character vector of additional fields to be included in the search results, e.g. `c("Station_Name", "Plant_Nickname")`.

`climate_data` Boolean value indicating that all climate variables should be included in `additional_fields`.

`ip_address` Optional field, string. IP Address of user requesting data. Used for generating data reports.

`dataset_ids` Integer vector of unique IDs for searching based on dataset, e.g. NEON or GRSM `c(17,15)`.

`email` Optional field, string. Email of user requesting data.

`download_path` Character, optional file path to the file for which to output the results.

`six_leaf_layer` Boolean value when set to `TRUE` will attempt to resolve the date of the observation to a spring index, leafing value for the location at which the observations was taken.

`six_bloom_layer`
Boolean value when set to `TRUE` will attempt to resolve the date of the observation to a spring index, bloom value for the location at which the observations was taken.

`agdd_layer` Numeric value, accepts `32` or `50`. When set, the results will attempt to resolve the date of the observation to an AGDD value for the location; the `32` or `50` represents the base value of the AGDD value returned. All AGDD values are based on a January 1st start date of the year in which the observation was taken.

six_sub_model
: Affects the results of the six layers returned. Can be used to specify one of three submodels used to calculate the spring index values. Thus setting this field will change the results of six_leaf_layer and six_bloom_layer. Valid values include: 'lilac', 'zabelli' and 'arnoldred'. For more information see the NPN's Spring Index Maps documentation: https://www.usanpn.org/data/maps/spring.

additional_layers
: Data frame with first column named name and containing the names of the layer for which to retrieve data and the second column named param and containing string representations of the time/elevation subset parameter to use. This variable can be used to append additional geospatial layer data fields to the results, such that the date of observation in each row will resolve to a value from the specified layers, given the location of the observation.

taxonomy_aggregate
: Boolean value indicating whether to aggregate data by a taxonomic order higher than species. This will be based on the values set in family_ids, order_ids, or class_ids. If one of those three fields are not set, then this value is ignored.

pheno_class_aggregate
: Boolean value indicating whether to aggregate data by the pheno class ids as per the pheno_class_ids parameter. If the pheno_class_ids value is not set, then this parameter is ignored. This can be used in conjunction with taxonomy_aggregate and higher taxonomic level data filtering.

wkt
: WKT geometry by which filter data. Specifying a valid WKT within the contiguous US will filter data based on the locations which fall within that WKT.

Details

This data type includes estimates of the overall onset and end of phenophase activity for plant and animal species at a site over a user-defined time period. Each row provides the first and last occurrences of a given phenophase on a given species, beginning with the date of the first observed "yes" phenophase status record and ending with the date of the last observed "yes" record of the user-defined time period. For plant species where multiple individuals are monitored at the site, the date provided for "first yes" is the mean of the first "yes" records for each individual plant at the site, and the date for "last yes" is the mean of the last "yes" records. Note that a phenophase may have ended and restarted during the overall period of its activity at the site. These more fine-scale patterns can be explored in the individual phenometrics data.

Most search parameters are optional, however, users are encouraged to supply additional search parameters to get results that are easier to work with. request_source must be provided. This is a self-identifying string, telling the service who is asking for the data or from where the request is being made. It is recommended you provide your name or organization name. If the call to this function is acting as an intermediary for a client, then you may also optionally provide a user email and/or IP address for usage data reporting later.

Additional fields provides the ability to specify additional, non-critical fields to include in the search results. A complete list of additional fields can be found in the NPN service's companion documentation. Metadata on all fields can be found in the following Excel sheet: https://www.usanpn.org/files/metadata/site_phenometrics_datafield_descriptions.xlsx

Value

A tibble of all status records returned as per the search parameters. If `download_path` is specified, the file path is returned instead.

Examples

```
## Not run:
#Download all saguaro data for 2013 and 2014
npn_download_site_phenometrics(
  request_source = "Your Name or Org Here",
  years = c(2013, 2014),
  species_id = 210,
  download_path = "saguaro_data_2013_2014.csv"
)

## End(Not run)
```

---

npn_download_status_data

Download Status and Intensity Records

Description

This function allows for a parameterized search of all status records in the USA-NPN database, returning all records as per the search parameters in a data table. Data fetched from NPN services is returned as raw JSON before being channeled into a data table. Optionally results can be directed to an output file in which case the raw JSON is converted to CSV and saved to file; in that case, data is also streamed to file which allows for more easily handling of the data if the search otherwise returns more data than can be handled at once in memory.

Usage

```
npn_download_status_data(
  request_source,
  years,
  coords = NULL,
  species_ids = NULL,
  genus_ids = NULL,
  family_ids = NULL,
  order_ids = NULL,
  class_ids = NULL,
  station_ids = NULL,
  species_types = NULL,
  network_ids = NULL,
  states = NULL,
  phenophase_ids = NULL,
  functional_types = NULL,
```

```
  additional_fields = NULL,
  climate_data = FALSE,
  ip_address = NULL,
  dataset_ids = NULL,
  email = NULL,
  download_path = NULL,
  six_leaf_layer = FALSE,
  six_bloom_layer = FALSE,
  agdd_layer = NULL,
  six_sub_model = NULL,
  additional_layers = NULL,
  pheno_class_ids = NULL,
  wkt = NULL
)
```

Arguments

request_source	Required field, character Self-identify who is making requests to the data service.
years	Required field, character vector. Specify the years to include in the search, e.g. c('2013','2014'). You must specify at least one year.
coords	Numeric vector, used to specify a bounding box as a search parameter, e.g. c(lower_left_lat, lower_left_long, upper_right, lat,upper_right_long).
species_ids	Integer vector of unique IDs for searching based on species, e.g. c(3, 34, 35).
genus_ids	Integer vector of unique IDs for searching based on taxonomic family, e.g. c(3, 34, 35). This parameter will take precedence if species_ids is also set.
family_ids	Integer vector of unique IDs for searching based on taxonomic family, e.g. c(3, 34, 35). This parameter will take precedence if species_ids is also set.
order_ids	Integer vector of unique IDs for searching based on taxonomic order, e.g. c(3, 34, 35). This parameter will take precedence if species_ids or family_ids are also set.
class_ids	Integer vector of unique IDs for searching based on taxonomic class, e.g. c(3, 34, 35). This parameter will take precedence if species_ids, family_ids or order_ids are also set.
station_ids	Integer vector of unique IDs for searching based on site location, e.g. c(5, 9).
species_types	Character vector of unique species type names for searching based on species types, e.g. c("Deciduous", "Evergreen").
network_ids	Integer vector of unique IDs for searching based on partner group/network, e.g. c(500, 300).
states	Character vector of US postal states to be used as search params, e.g. c("AZ", "IL").
phenophase_ids	Integer vector of unique IDs for searching based on phenophase, e.g. c(323, 324).
functional_types	Character vector of unique functional type names, e.g. ‘c("Birds").

additional_fields
: Character vector of additional fields to be included in the search results, e.g. `c("Station_Name", "Plant_Nickname")`.

climate_data
: Boolean value indicating that all climate variables should be included in `additional_fields`.

ip_address
: Optional field, string. IP Address of user requesting data. Used for generating data reports.

dataset_ids
: Integer vector of unique IDs for searching based on dataset, e.g. NEON or GRSM `c(17,15)`.

email
: Optional field, string. Email of user requesting data.

download_path
: Character, optional file path to the file for which to output the results.

six_leaf_layer
: Boolean value when set to `TRUE` will attempt to resolve the date of the observation to a spring index, leafing value for the location at which the observations was taken.

six_bloom_layer
: Boolean value when set to `TRUE` will attempt to resolve the date of the observation to a spring index, bloom value for the location at which the observations was taken.

agdd_layer
: Numeric value, accepts `32` or `50`. When set, the results will attempt to resolve the date of the observation to an AGDD value for the location; the `32` or `50` represents the base value of the AGDD value returned. All AGDD values are based on a January 1st start date of the year in which the observation was taken.

six_sub_model
: Affects the results of the six layers returned. Can be used to specify one of three submodels used to calculate the spring index values. Thus setting this field will change the results of `six_leaf_layer` and `six_bloom_layer`. Valid values include: `'lilac'`, `'zabelli'` and `'arnoldred'`. For more information see the NPN's Spring Index Maps documentation: https://www.usanpn.org/data/maps/spring.

additional_layers
: Data frame with first column named `name` and containing the names of the layer for which to retrieve data and the second column named `param` and containing string representations of the time/elevation subset parameter to use. This variable can be used to append additional geospatial layer data fields to the results, such that the date of observation in each row will resolve to a value from the specified layers, given the location of the observation.

pheno_class_ids
: Integer vector of unique IDs for searching based on pheno class. Note that if both `pheno_class_id` and `phenophase_id` are provided in the same request, `phenophase_id` will be ignored.

wkt
: WKT geometry by which filter data. Specifying a valid WKT within the contiguous US will filter data based on the locations which fall within that WKT.

Details

Most search parameters are optional. However, users are encouraged to supply additional search parameters to get results that are easier to work with. `request_source` must be provided. This is a self-identifying string, telling the service who is asking for the data or from where the request is

being made. It is recommended you provide your name or organization name. If the call to this function is acting as an intermediary for a client, then you may also optionally provide a user email and/or IP address for usage data reporting later.

Additional fields provides the ability to specify more, non-critical fields to include in the search results. A complete list of additional fields can be found in the NPN service's companion documentation. Metadata on all fields can be found in the following Excel sheet: https://www.usanpn.org/files/metadata/status_intensity_datafield_descriptions.xlsx

Value

A tibble of all status records returned as per the search parameters. If download_path is specified, the file path is returned instead.

Examples

```
## Not run:
#Download all saguaro data for 2016
npn_download_status_data(
  request_source = "Your Name or Org Here",
  years = c(2016),
  species_id = c(210),
  download_path = "saguaro_data_2016.csv"
)

## End(Not run)
```

---

npn_get_agdd_point_data

Get AGDD Point Value

Description

This function is for requesting AGDD point values. Because the NPN has a separate data service that can provide AGDD values which is more accurate than Geoserver this function is ideal when requested AGDD point values.

Usage

```
npn_get_agdd_point_data(layer, lat, long, date, store_data = TRUE)
```

Arguments

layer	The name of the queried layer.
lat	The latitude of the queried point.
long	The longitude of the queried point.
date	The queried date.
store_data	Boolean value. If set TRUE then the value retrieved will be stored in a global variable named point_values for later use.

Details

As this function only works for AGDD point values, if it's necessary to retrieve point values for other layers please try the `npn_get_point_data()` function.

Value

Returns a numeric value of the AGDD value at the specified lat/long/date. If no value can be retrieved, then `-9999` is returned.

Examples

```
## Not run:
npn_get_agdd_point_data(
  layer = "gdd:agdd",
  lat = 32.4,
  long = -110,
  date = "2020-01-15"
)

## End(Not run)
```

`npn_get_custom_agdd_raster`

Get Custom AGDD Raster Map

Description

This function takes a series of variables used in calculating AGDD and returns a raster of the continental USA with each pixel representing the calculated AGDD value based on start and end date. This function leverages the USA-NPN geo web services.

Usage

```
npn_get_custom_agdd_raster(
  method,
  climate_data_source,
  temp_unit,
  start_date,
  end_date,
  base_temp,
  upper_threshold = NULL
)
```

Arguments

method	Takes "simple" or "double-sine" as input. This is the AGDD calculation method to use for each data point. Simple refers to simple averaging.
climate_data_source	Specified the climate data set to use. Takes either "PRISM" or "NCEP" as input.
temp_unit	The unit of temperature to use in the calculation. Takes either "Fahrenheit" or "Celsius" as input.
start_date	Date at which to begin the AGDD calculations.
end_date	Date at which to end the AGDD calculations.
base_temp	This is the lowest temperature for each day for it to be considered in the calculation.
upper_threshold	This parameter is only applicable for the double-sine method. This sets the highest temperature to be considered in any given day's AGDD calculation.

Value

A terra::SpatRaster object of each calculated AGDD numeric values based on specified time period/method/base temp/data source.

Examples

```
## Not run:
res <- npn_get_custom_agdd_raster(
  method = "simple",
  climate_data_source = "NCEP",
  temp_unit = "Fahrenheit",
  start_date = "2020-01-01",
  end_date = "2020-01-15",
  base_temp = 32
)

## End(Not run)
```

---

npn_get_custom_agdd_time_series

Get Custom AGDD Time Series

Description

This function takes a series of variables used in calculating AGDD and returns an AGDD time series, based on start and end date, for a given location in the continental US. This function leverages the USA-NPN geo web services

Usage

```
npn_get_custom_agdd_time_series(
  method,
  start_date,
  end_date,
  base_temp,
  climate_data_source,
  temp_unit,
  lat,
  long,
  upper_threshold = NULL
)
```

Arguments

method	Takes "simple" or "double-sine" as input. This is the AGDD calculation method to use for each data point. Simple refers to simple averaging.
start_date	Date at which to begin the AGDD calculations.
end_date	Date at which to end the AGDD calculations.
base_temp	This is the lowest temperature for each day for it to be considered in the calculation.
climate_data_source	Specified the climate data set to use. Takes either "PRISM" or "NCEP" as input.
temp_unit	The unit of temperature to use in the calculation. Takes either "Fahrenheit" or "Celsius" as input.
lat	The latitude of the location for which to calculate the time series.
long	The longitude of the location for which to calculate the time series.
upper_threshold	This parameter is only applicable for the double-sine method. This sets the highest temperature to be considered in any given day's AGDD calculation.

Value

A data frame containing the numeric AGDD values for each day for the specified time period/location/method/base temp/data source.

Examples

```
## Not run:
res <- npn_get_custom_agdd_time_series(
  method = "double-sine",
  start_date = "2019-01-01",
  end_date = "2019-01-15",
  base_temp = 25,
  climate_data_source = "NCEP",
  temp_unit = "fahrenheit",
  lat = 39.7,
```

```
  long = -107.5,
  upper_threshold = 90
)

## End(Not run)
```

`npn_get_layer_details` *Get Geospatial Data Layer Details*

Description

This function will return information about the various data layers available via the NPN's geospatial web services. Specifically, this function will query the NPN's GetCapabilities endpoint and parse the information on that page about the layers. For each layer, this function will retrieve the layer name (as to be specified elsewhere programmatically), the title (human readable), the abstract, which describes the data in the layer, the dimension name and dimension range for specifying specific date values from the layer.

Usage

```
npn_get_layer_details()
```

Details

Information about the layers can also be viewed at the getCapbilities page directly: https://geoserver.usanpn.org/geoserver/wms?request=GetCapabilities

Value

A tibble containing all layer details as specified in function description.

Examples

```
## Not run:
layers <- npn_get_layer_details()

## End(Not run)
```

`npn_get_phenophases_for_taxon` *Get Phenophases for Taxon*

Description

This function gets a list of phenophases that are applicable for a provided taxonomic grouping, e.g. family, order. Note that since a higher taxononmic order will aggregate individual species not every phenophase returned through this function will be applicable for every species belonging to that taxonomic group.

Usage

```
npn_get_phenophases_for_taxon(
  family_ids = NULL,
  order_ids = NULL,
  class_ids = NULL,
  genus_ids = NULL,
  date = NULL,
  return_all = 0,
  ...
)
```

Arguments

`family_ids`	Integer vector of taxonomic family ids to search for.
`order_ids`	Integer vector of taxonomic order ids to search for.
`class_ids`	Integer vector of taxonomic class ids to search for
`genus_ids`	Integer vector of taxonomic genus ids to search for
`date`	Specify the date of interest. For this function to return anything, either this value must be set or `return_all` must be `1`.
`return_all`	Takes either `0` or `1` as input and defaults to `0`. For this function to return anything, either this value must be set to `1` or `date` must be set.
`...`	Currently unused.

Details

It's also important to note that phenophase definitions can change for individual species over time, so there's a need to specify either a date of interest, or to explicitly state that the function should return all phenophases that were ever applicable for any species belonging to the specified taxonomic group.

When called, this function requires of these three parameters, exactly one of `family_ids`, `order_ids` or `class_ids` to be set.

Value

A data frame listing phenophases in the NPN database for the specified taxon and date.

Examples

```
## Not run:
npn_get_phenophases_for_taxon(class_ids = c(5, 6), date = "2018-05-05")
npn_get_phenophases_for_taxon(family_ids = c(267, 268), date = "2018-05-05")

#if you supply two or more "ids" arguments, the highest classification takes precedence
pheno <- npn_get_phenophases_for_taxon(
  class_ids = 4,
  family_ids = c(103, 104),
  genus_ids = c(409, 957, 610),
  date = "2018-05-05"
)

colnames(pheno)
# [1] "family_id"   "family_name" "phenophases"

## End(Not run)
```

---

npn_get_point_data *Get Point Data Value*

Description

This function can get point data about any of the NPN geospatial layers.

Usage

```
npn_get_point_data(layer, lat, long, date, store_data = TRUE)
```

Arguments

layer	The coverage id (machine name) of the layer for which to retrieve. Applicable values can be found via the npn_get_layer_details() function under the name column.
lat	The latitude of the point.
long	The longitude of the point.
date	The date for which to get a value.
store_data	Boolean value. If set TRUE then the value retrieved will be stored in a global variable named point_values for later use.

Details

Please note that this function pulls this from the NPN's WCS service so the data may not be totally precise. If you need precise AGDD values try using the npn_get_agdd_point_data() function.

Value

Returns a numeric value for any NPN geospatial data layer at the specified lat/long/date. If no value can be retrieved, then `-9999` is returned.

Examples

```
## Not run:
value <-
  npn_get_point_data(
    layer = "gdd:agdd",
    lat = 38.8,
    long = -110.5,
    date = "2022-05-05"
  )

## End(Not run)
```

`npn_groups` *Get Partner Groups*

Description

Returns a list of all groups participating in the NPN's data collection program. These details can be used to further filter other service endpoints' results.

Usage

```
npn_groups(use_hierarchy = FALSE, ...)
```

Arguments

`use_hierarchy`	Boolean indicating whether or not the list of networks should be represented in a hierarchy. If `TRUE`, the result will be returned as a nested list rather than a tibble. Defaults to `FALSE`.
`...`	Currently unused.

Value

A tibble (or nested list if `use_hierarchy = TRUE`) of partner groups, including `network_id` and `network_name`.

Examples

```
## Not run:
npn_groups()
npn_groups(use_heirarchy = TRUE)

## End(Not run)
```

npn_lookup_names *Species Name Lookup*

Description

Look up species IDs by taxonomic or common name

Usage

```
npn_lookup_names(name, type = "genus", fuzzy = FALSE)
```

Arguments

name	A scientific or common name.
type	One of "common_name", "genus", or "species".
fuzzy	Logical; if TRUE, uses fuzzy search via agrep(), if FALSE, uses grep().

Value

A data frame with species ID numbers based on the name and type parameters.

Examples

```
## Not run:
npn_lookup_names(name = 'Pinus', type = 'genus')
npn_lookup_names(name = 'pine', type = 'common_name')
npn_lookup_names(name = 'bird', type = 'common_name', fuzzy = TRUE)

## End(Not run)
```

npn_phenophases *Get Phenophases*

Description

Retrieves a complete list of all phenophases in the NPN database.

Usage

```
npn_phenophases(...)
```

Arguments

...	Currently unused.

Value

A tibble listing all phenophases available in the NPN database.

Examples

```
## Not run:
phenophases <- npn_phenophases()

## End(Not run)
```

`npn_phenophases_by_species`
Get Phenophase for Species

Description

Retrieves the phenophases applicable to species for a given date. It's important to specify a date since protocols/phenophases for any given species can change from year to year.

Usage

```
npn_phenophases_by_species(species_ids, date, ...)
```

Arguments

`species_ids`	Integer vector of species IDs for which to get phenophase information.
`date`	The applicable date for which to retrieve phenophases for the given species.
`...`	Currently unused.

Value

A tibble listing phenophases in the NPN database for the specified species and date.

Examples

```
## Not run:
pp <- npn_phenophases_by_species(3, "2018-05-05")

## End(Not run)
```

npn_phenophase_definitions *Get Phenophase Definitions*

Description

Retrieves a complete list of all phenophase definitions.

Usage

```
npn_phenophase_definitions(...)
```

Arguments

...	Currently unused.

Value

A tibble listing all phenophases in the NPN database and their definitions.

Examples

```
## Not run:
pp <- npn_phenophase_definitions()

## End(Not run)
```

npn_phenophase_details *Get Phenophase Details*

Description

Retrieves additional details for select phenophases, including full list of applicable phenophase definition IDs and phenophase revision notes over time

Usage

```
npn_phenophase_details(ids = NULL, ...)
```

Arguments

ids	Takes a vector of phenophase ids for which to retrieve additional details.
...	Currently unused.

Value

A tibble listing phenophases in the NPN database, including detailed information for each, filtered by the phenophase ID.

Examples

```
## Not run:
pd <- npn_phenophase_details(c(56, 57))

## End(Not run)
```

`npn_pheno_classes` *Get Pheno Classes*

Description

Gets information about all pheno classes, which are a higher-level order of phenophases.

Usage

```
npn_pheno_classes(...)
```

Arguments

`...`	Currently unused.

Value

A tibble listing the pheno classes in the NPN database.

Examples

```
## Not run:
pc <- npn_pheno_classes()

## End(Not run)
```

npn_set_env *Set Environment*

Description

By default this library will call the NPN's production services but in some cases it's preferable to access the development web services so this function allows for manually setting the web service endpoints to use DEV instead. Just pass in "dev" to this function to change the endpoints to use.

Usage

```
npn_set_env(env = "ops")
```

Arguments

env	The environment to use. Should be "ops" or "dev"

npn_species *Get Species*

Description

Returns a complete list of all species information of species represented in the NPN database.

Returns information about a species based on the NPN's unique ID for that species

Search for species by state

Search NPN species information using a number of different parameters, which can be used in conjunction with one another, including:

- Species on which a particular group or groups are actually collecting data
- What species were observed in a given date range
- What species were observed at a particular station or stations

Usage

```
npn_species(...)

npn_species_id(ids, ...)

npn_species_state(state, kingdom = NULL, ...)

npn_species_search(
  network = NULL,
  start_date = NULL,
  end_date = NULL,
  station_id = NULL,
  ...
)
```

Arguments

...	Currently unused.
ids	Integer vector of species ids for which to retrieve information.
state	A US postal state code to filter results.
kingdom	Filters results by taxonomic kingdom. Valid values include 'Animalia', 'Plantae'.
network	filter species based on identifiers of NPN groups that are actually observing data on the species. Takes a single numeric ID.
start_date	filter species by date observed. This sets the start date of the date range and must be used in conjunction with end_date.
end_date	filter species by date observed. This sets the end date of the date range and must be used in conjunction with start_date.
station_id	filter species by a numeric vector of unique site identifiers.

Value

A tibble with information on species in the NPN database and their IDs.

A tibble with information on species in the NPN database and their IDs, filtered by the species ID parameter.

A tibble with information on species in the NPN database whose distribution includes a given state.

A tibble with information on species in the NPN database filtered by partner group, dates and station/site IDs.

Examples

```
## Not run:
npn_species()

## End(Not run)
## Not run:
npn_species_id(ids = 3)

## End(Not run)
## Not run:
npn_species_state(state = "AZ")
npn_species_state(state = "AZ", kingdom = "Plantae")

## End(Not run)
## Not run:
species <- npn_species_search(
  start_date = "2013-01-01",
  end_date = "2013-05-15"
)

## End(Not run)
```

npn_species_types *Get Species Types*

Description

Return all plant or animal functional types used in the NPN database.

Usage

```
npn_species_types(kingdom = "Plantae", ...)
```

Arguments

kingdom	Filters results by taxonomic kingdom. Valid values include 'Animalia', 'Plantae', or NULL (which returns results for both). Defaults to 'Plantae'.
...	Currently unused.

Value

A data frame with a list of the functional types used in the NPN database, filtered by the specified kingdom.

Examples

```
## Not run:
npn_species_types("Plantae")

## End(Not run)
```

npn_stations *Get Station Data*

Description

Get a list of all stations, optionally filtered by state

Usage

```
npn_stations(state_code = NULL, ...)
```

Arguments

state_code	The postal code of the US state by which to filter the results returned. Leave empty to get all stations.
...	Currently unused.

Value

A data frame with stations' latitude and longitude, names, and ids.

Examples

```
## Not run:
npn_stations()
npn_stations('AZ')

## End(Not run)
```

`npn_stations_by_location` *Get station data based on a WKT defined geography.*

Description

Takes a Well-Known Text based geography as input and returns data for all stations, including unique IDs, within that boundary.

Usage

```
npn_stations_by_location(wkt, ...)
```

Arguments

`wkt`	Required field specifying the WKT geography to use.
`...`	Currently unused.

Value

A data frame listing stations filtered based on the WKT geography.

Examples

```
## Not run:
head(npn_stations_by_state(wkt = "POLYGON((
-110.94484396954107 32.23623109416672,-110.96166678448247 32.23594069208043,
-110.95960684795904 32.21328646993733,-110.94244071026372 32.21343170728929,
-110.93935080547857 32.23216538049456,-110.94484396954107 32.23623109416672))")
)

## End(Not run)
```

npn_stations_by_state *Get number of stations by state.*

Description

Get number of stations by state.

Usage

```
npn_stations_by_state(...)
```

Arguments

...	Currently unused.

Value

A data frame listing stations by state.

Examples

```
## Not run:
head(npn_stations_by_state())

## End(Not run)
```

npn_stations_with_spp *Get Stations with Species*

Description

Get a list of all stations which have an individual whom is a member of a set of species.

Usage

```
npn_stations_with_spp(species_id, ..., speciesid = deprecated())
```

Arguments

species_id	Required. Species id numbers, from 1 to infinity, potentially, use e.g., c(52, 53), if more than one species desired (numeric).
...	Currently unused.
speciesid	Deprecated. Use species_id instead.

Value

A data frame with stations' latitude and longitude, names, and ids.

Examples

```
## Not run:
npn_stations_with_spp(species_id = c(52, 53, 54))
npn_stations_with_spp(species_id = 53)

## End(Not run)
```

`rnpn-defunct` *Defunct functions in rnpn*

Description

- `npn_obsspbyday()`: Removed.
- `npn_allobssp()`: Removed.
- `npn_indspatstations()`: Removed.
- `npn_indsatstations()`: Removed.
- `npn_stationsbystate()`: Removed.
- `npn_stationswithspp()`: Removed.

Index

∗ **package**
    rnpn-package, 2

agrep(), *27*

grep(), *27*

npn_abundance_categories, 3
npn_allobssp(), *36*
npn_datasets, 4
npn_download_geospatial, 4
npn_download_individual_phenometrics, 5
npn_download_magnitude_phenometrics, 9
npn_download_site_phenometrics, 12
npn_download_status_data, 16
npn_get_agdd_point_data, 19
npn_get_agdd_point_data(), *25*
npn_get_custom_agdd_raster, 20
npn_get_custom_agdd_time_series, 21
npn_get_layer_details, 23
npn_get_layer_details(), *5*, *25*
npn_get_phenophases_for_taxon, 24
npn_get_point_data, 25
npn_get_point_data(), *20*
npn_groups, 26
npn_indsatstations(), *36*
npn_indspatstations(), *36*
npn_lookup_names, 27
npn_obsspbyday(), *36*
npn_pheno_classes, 30
npn_phenophase_definitions, 29
npn_phenophase_details, 29
npn_phenophases, 27
npn_phenophases_by_species, 28
npn_set_env, 31
npn_species, 31
npn_species_id *(*npn_species*)*, 31
npn_species_search *(*npn_species*)*, 31
npn_species_state *(*npn_species*)*, 31
npn_species_types, 33
npn_stations, 33
npn_stations_by_location, 34
npn_stations_by_state, 35
npn_stations_with_spp, 35
npn_stationsbystate(), *36*
npn_stationswithspp(), *36*

rnpn *(*rnpn-package*)*, 2
rnpn-defunct, 36
rnpn-package, 2

terra::SpatRaster, *5*, *21*