Package: NSR (via r-universe)

October 22, 2026

Title 'Native Status Resolver'

Version 0.1.1

Description Provides access to the 'Native Status Resolver' (NSR) <https://github.com/ojalaquellueva/nsr> API through R. The user supplies plant taxonomic names and political divisions and the package returns information about their likely native status (e.g., native, non-native,endemic), along with information on how those decisions were made.

Depends R (>= 3.5.0)

License MIT + file LICENSE

Encoding UTF-8

LazyData true

Imports jsonlite, httr, stats, tools, utils

Suggests knitr, data.table, nanoparquet, rmarkdown, rWCVPdata, sf, terra, testthat, devtools, TNRS, vcr (>= 0.6.0), withr

Additional_repositories https://matildabrown.github.io/drat

VignetteBuilder knitr, rmarkdown

RoxygenNote 7.3.3

NeedsCompilation no

Author Brian Maitner [aut, cre] (ORCID: <https://orcid.org/0000-0002-2118-9880>), Brad Boyle [aut]

Maintainer Brian Maitner <bmaitner@gmail.com>

Config/pak/sysreqs libssl-dev

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

Date/Publication 2026-10-21 19:26:03 UTC

RemoteUrl https://github.com/cran/NSR

RemoteRef HEAD

RemoteSha 39ba63e8fe3e95f8d040624adb2a6f506ce129d0

Contents

NSR . . . 2
NSR_citations . . . 3
NSR_data_dictionary . . . 4
NSR_local . . . 5
NSR_local_build . . . 7
NSR_local_by_region . . . 8
NSR_local_remove . . . 9
NSR_local_status . . . 10
NSR_metadata . . . 10
NSR_political_divisions . . . 11
NSR_simple . . . 12
NSR_sources . . . 13
NSR_template . . . 14
nsr_testfile . . . 14
NSR_version . . . 15

Index **16**

`NSR` *Check the native status for plant species in a political region*

Description

NSR returns information on native status for species within a political region.

Usage

```
NSR(occurrence_dataframe, ...)
```

Arguments

`occurrence_dataframe`
: A properly formatted dataframe, see https://bien.nceas.ucsb.edu/bien/tools/nsr/batch-mode/

`...`
: Additional arguments passed to internal functions.

Details

The NSR API accepts at most 5,000 observations per call. Larger inputs are split into batches of at most 5,000 rows, sent separately, and recombined in the order of the input.

Value

Dataframe containing NSR results.

Examples

```
## Not run:

results <- NSR(occurrence_dataframe = nsr_testfile)

# Inspect the results
head(results, 10)
# That's a lot of columns. Let's display one row vertically
# to get a better understanding of the output fields
results.t <- t(results[,2:ncol(results)])
results.t[,1,drop =FALSE]
# Summarize the main results
results[ 1:10,
c("species", "country", "state_province", "native_status", "native_status_reason")]

# Compare summary flag isIntroduced to more detailed native_status values
# and inspect souces consulted
results[ 1:10,
c("species", "country", "state_province", "native_status", "isIntroduced", "native_status_sources")]





## End(Not run)
```

`NSR_citations` *Get citation information*

Description

Returns information needed to cite the NSR

Usage

```
NSR_citations(...)
```

Arguments

`...`	Additional arguments passed to internal functions.

Value

Dataframe containing bibtex-formatted citation information

Note

This function provides citation information in bibtex format that can be used with reference manager software (e.g. Paperpile, Zotero). Please do remember to cite both the sources and the NSR, as the NSR couldn't exist without these sources!

Examples

```
{
citation_info <- NSR_citations()
}
```

NSR_data_dictionary *Get NSR data dictionary*

Description

Returns information from the NSR data dictionary

Usage

```
NSR_data_dictionary(native_status = FALSE, ...)
```

Arguments

native_status	Logical. If FALSE(Default) returns information on fields. If TRUE, returns information on Native Status categories.
...	Additional arguments passed to internal functions.

Value

Data.frame containing requested data dictionary contents.

Examples

```
{
NSR_fields <- NSR_data_dictionary()

status_codes <- NSR_data_dictionary(native_status = TRUE)
}
```

NSR_local *Determine native status without an internet connection*

Description

Offline Native Status Resolver. For each taxon and place, the status every built checklist gives it, reduced to one answer.

Usage

```
NSR_local(
  occurrence_dataframe,
  dir = nsr_cache_dir(),
  resolve_names = TRUE,
  use_coordinates = FALSE,
  min_overlap = 0.01,
  exclude_extinct = TRUE,
  quiet = FALSE
)
```

Arguments

`occurrence_dataframe`
: A data.frame with `species`, and either coordinates or political division names (see Place).

`dir`
: Cache directory, shared with GNRS and GVS.

`resolve_names`
: Resolve submitted names against WCVP with `TNRS::TNRS_local()`? Names already matching WCVP accepted names need none.

`use_coordinates`
: Place each record by its `latitude` and `longitude` as well as by its division names? Default `FALSE`: coordinates cost a raster lookup per call, and resolving them to political divisions is GVS's job. See Place.

`min_overlap`
: Ignore region links covering less than this share of a region.

`exclude_extinct`
: Ignore distribution records the source marks extinct? Default `TRUE`, which answers about the present day. Set `FALSE` to model a past distribution, where a region the taxon has since been lost from still counts. See Extinct records.

`quiet`
: Suppress progress messages?

Details

Output carries NSR's columns, and adds: `isEndemic`; `native_status_conflict` and `native_status_conflict_type`, saying whether and how the sources disagreed; `native_status_scope`, which of the polygons bearing on the place produced the answer; `n_subpolygons_native` and `n_subpolygons_introduced`; `native_status_opinions`, every opinion consulted; `native_status_coordinates`, `native_status_names` and `place_conflict` (see Place); `regions_matched`, what the place was matched on; and `taxon_evaluable`,

whether any source holds native status for the taxon at all - which is what separates A (absent) from UNK (no data). The coordinates given are echoed as latitude and longitude.

Place. By default the answer comes from political division names (country, state_province, county_parish), which are resolved to GADM units through the GNRS backbone and carried to each source's own geography by the spatial link table. Any latitude and longitude present are echoed but not used: placing every record by point costs a raster lookup per call, and turning coordinates into political divisions is what GVS already does. Resolve them there and pass the divisions here, or set use_coordinates = TRUE to have this function do it.

With use_coordinates = TRUE, a record is placed by its point in each source's own geography (WCVP against WGSRPD level-3 areas, VASCAN and Flora do Brasil against GADM states), with no crosswalk between them - the finer question, where the data supports it. That needs both rasters. If one is missing, the point is placed in the other geography and its sources are reached through the region link table instead - the route a division name takes, and no more precise than one; a warning says so, and NSR_local_status() reports it as coordinate_route = "crosswalk". Given coordinates and names together, the two are independent claims about where the record is and either can be wrong: a transposed longitude and a mistyped province are equally easy mistakes, so neither is authoritative. Where they agree, both are used. Where they disagree, place_conflict is TRUE, native_status is UNK, and the two bases are answered separately in native_status_coordinates and native_status_names for you to judge. The service has no combined mode to follow here: NSR takes division names only.

Precedence. If any source says native, the answer is native: asserting nativity is a positive claim, whereas "introduced" and "absent" are often artefacts of a list's scope, age or purpose. Disagreement is recorded, not hidden, in native_status_conflict and native_status_opinions.

Rank. A distribution recorded against an accepted infraspecific taxon counts for its species: a subspecies of *X* native somewhere means *X* is native there. Downward, a genus (or family) speaks for a species only through claims true of every species in it, and only where the species has no record of its own bearing on the place. A genus native to a region says nothing about which of its species are - it is the introduced one as readily as the native one - so it never makes a species native. But a genus introduced in a polygon that is or contains the place has no native species there, so the species is I; and a genus absent from the place's comprehensive lists, with its whole native range confined elsewhere, makes it Ie. The reason names the higher taxon, and native_status_scope is "genus" or "family". Only native_status is inferred this way; the per-level codes are the species' own records. A genus *query* is answered directly, from the distributions WCVP records against genera. The live service makes the second inference but not the first, and also calls a species Ne where its genus is endemic to the place, which NSR_local() does not.

Which polygons answer. A place is judged by the polygons it lies IN. An opinion about a polygon containing the place applies to it (POWO's finest statement about Guadeloupe is "native in the Leeward Islands"), and among those any native opinion wins. Polygons INSIDE the place describe only parts of it, so they are not its status: they answer only when they all agree, and otherwise the answer is P with the reason saying the status varies and how many sub-polygons say what (n_subpolygons_native, n_subpolygons_introduced). Give coordinates and the question does not arise: the record is judged on the ground it sits on. native_status_scope records which of these produced the answer.

Endemism is the exception, deliberately: Ne and Ie are claims about the taxon's whole range rather than about one polygon, so they draw on evidence from elsewhere. A record of a taxon confined to California, found in Michigan, is introduced there however little Michigan's own checklists say.

Value

A data.frame, one row per input row, carrying `user_id` from the input (or sequential ids where the input has none), as `NSR` does.

Extinct records

WCVP marks a distribution extinct with a bare flag and no date, so the record says only "considered no longer present as of this release". Whether it should count is therefore a property of the question rather than of the data: a present-day status wants it out, a historical distribution wants it in, and the cache keeps both so `exclude_extinct` can decide per call.

Excluding it never makes the taxon introduced there. The endemism rules read the taxon's whole native range, extinct records included, whatever this argument says - a region a taxon has been lost from is still a region it was native to, so `Ie` ("introduced, inferred from endemism elsewhere") cannot fire against it. The answer for such a record is `A`: gone, not foreign.

County and parish

`county_parish` is accepted and echoed, but not resolved: the political-division backbone stops at state and province, so `native_status_county_parish` is always `NA` and the answer is given at the finest level that could be matched. A warning says so. Coordinates are the way to ask a finer question.

`NSR_local_build` *Build the local native-status reference*

Description

Reads each checklist source, resolves its names against WCVP with `TNRS::TNRS_local()` and its political divisions against the GNRS backbone, and writes the shared cache tables. Sources are acquired on the user's machine; nothing derived ships with the package.

Usage

```
NSR_local_build(
  sources = "wcvp",
  dir = nsr_cache_dir(create = TRUE),
  files = list(),
  overwrite = FALSE,
  quiet = FALSE
)
```

Arguments

`sources`	Which sources to build. See `NSR_local_status()`.
`dir`	Cache directory, shared with GNRS and GVS.

`files`	Named list of local archives to read, e.g. `list(flbr = "flbr_dwca.zip")`. Required for `vascan` and `flbr`. For `wcvp`, the WCVP zip; if absent, the copy in the TNRS cache is used when there is one.
`overwrite`	Rebuild sources that are already built?
`quiet`	Suppress progress messages?

Details

The archives are not downloaded for you. `vascan` and `flbr` must be supplied through `files`, and each stops with the URL to fetch if they are not; `wcvp` is the exception, since it reads the WCVP archive already in the TNRS cache when there is one. This is why `sources` defaults to `"wcvp"` alone: it is the only source that can build unattended.

`"powo"` is accepted as a synonym for `"wcvp"` - it is the name the live service reports for the same Kew dataset - but `wcvp` is what gets written to `native_status_sources`.

Tables written: `nsr-sources` (one per source, with licence and whether it is comprehensive), `nsr-taxa` (one per taxon, keyed on the WCVP accepted id), `nsr-regions` (one per region, in the geography its source publishes against) and `nsr-checklist` (one per taxon x region x source). `nsr-region-links`, the spatial relations between geographies, is built too when the GADM index from GVS is in the cache.

Answering by division name needs the GNRS backbone in the same cache, since names reach every source through GADM; build it with GNRS. `vascan` and `flbr` need it to build at all. A `wcvp`-only build does not, and without it answers by coordinates only - `NSR_local_status()` reports which.

Value

Invisibly, `NSR_local_status()`.

`NSR_local_by_region` *Native status for taxa and polygons you have already resolved*

Description

A pipeline that has located its records itself - in the WGSRPD raster and the GADM index, as an occurrence workflow does once for all of its coordinates - already knows the polygons each record sits in. This takes those directly, skipping name resolution and point-in-polygon, and answers with the same rules as `NSR_local`.

Usage

```
NSR_local_by_region(
  taxon_id,
  region_keys,
  country_keys = NULL,
  dir = nsr_cache_dir(),
  min_overlap = 0.01,
  exclude_extinct = TRUE
)
```

Arguments

taxon_id	WCVP accepted ids (character), one per query.
region_keys	The polygons each record sits in, as a list of character vectors ("wgsrpd3:BZL", "gadm1:BRA.25_1"), or a single character vector for one polygon each.
country_keys	Optional country polygons ("gadm0:BRA"), same shape.
dir	Cache directory.
min_overlap	Ignore region links covering less than this share of a region.
exclude_extinct	Ignore distribution records the source marks extinct? See NSR_local.

Value

A data.frame with native_status and the status columns NSR_local() returns, keyed on taxon_id. It does not echo the input columns NSR_local() carries through (species, the division names, user_id), since this entry point is given ids and polygon keys rather than a record.

NSR_local_remove *Remove the local NSR tables*

Description

Remove the local NSR tables

Usage

```
NSR_local_remove(dir = nsr_cache_dir(), quiet = FALSE)
```

Arguments

dir	Cache directory.
quiet	Suppress messages?

Value

Invisibly, the files removed.

NSR_local_status *Which parts of the local NSR are built*

Description

Which parts of the local NSR are built

Usage

```
NSR_local_status(dir = nsr_cache_dir())
```

Arguments

dir	Cache directory. Defaults to the one shared with GNRS and GVS.

Value

A data.frame with one row per source: whether it is built, how many checklist records and taxa it contributed, and when it was built; and whether a query can actually reach it - `answers_by_name` and `answers_by_coordinates` - and by which route coordinates get there, `coordinate_route`: `"direct"` when the point is placed in the source's own geography, `"crosswalk"` when it is placed only in the other geography and carried over by the region link table (the route a division name takes, and no more precise), `"none"` otherwise. A source can be built and still unreachable: WCVP's opinions are published against WGSRPD areas, so names reach them only through the link table, and coordinates directly only through the WGSRPD raster. If either is missing, `built` is `TRUE` and these columns say what is lost; re-run `NSR_local_build(overwrite = TRUE)` and read its warnings for the cause.

NSR_metadata *Get NSR metadata*

Description

Returns metadata on NSR including version and citation information

Usage

```
NSR_metadata(bibtex_file = NULL, ...)
```

Arguments

bibtex_file	Optional output file for writing bibtex citations.
...	Additional arguments passed to internal functions.

Value

List containing: (1) bibtex-formatted citation information, (2) information about NSR data sources, and (3) NSR version information. If any of these can't be retrieved (e.g. the API is unreachable), a message is given and NULL is returned.

Note

This function provides citation information in bibtex format that can be used with reference manager software (e.g. Paperpile, Zotero). Please remember to cite both the sources and the NSR, as the NSR couldn't exist without these sources!

This function is a wrapper that returns the output of the functions NSR_citations, NSR_sources, and NSR_version.

Examples

```
{
metadata <- NSR_metadata()
}
```

NSR_political_divisions

Get information on political divisions with checklists within the NSR

Description

NSR_political_divisions returns information on political divisions with checklist information present in the NSR.

Usage

```
NSR_political_divisions(by_country = TRUE, ...)
```

Arguments

by_country	Logical. If TRUE (the default), will return a data.frame of checklists for each country. If FALSE, will return a data.frame of countries for each checklist.
...	Additional parameters passed to internal functions.

Value

data.frame containing information on political divisions within the NSR database.

Note

Setting checklist to FALSE returns a list of political divisions that can be used to standardize spellings.

Examples

```
## Not run:

#To get a list of all political divisions with comprehensive checklists:
checklists_per_country <- NSR_political_divisions()

#To get a list of all checklists the associated countries, set "by_country" to FALSE
countries_per_checklist <- NSR_political_divisions(by_country=FALSE)


## End(Not run)
```

NSR_simple *Check the native status for plant species in a political region*

Description

NSR_simple returns information on native status for species within a political region.

Usage

```
NSR_simple(
  species = NULL,
  country = NULL,
  state_province = NULL,
  county_parish = NULL,
  ...
)
```

Arguments

species	A single species or a vector of species, with genus and specific epithet separated by a space.
country	A single country or a vector of countries. If a vector, length must equal length of species vector.
state_province	A single state/province or a vector of states. If a vector, length must equal length of species vector.
county_parish	A single county/parish or a vector of counties. If a vector, length must equal length of species vector.
...	Additional arguments passed to internal functions.

Value

Dataframe containing NSR results.

Examples

```
## Not run:

results <- NSR_simple(species = "Acer rubrum",
           country = "Canada",state_province = "Ontario")

results <- NSR_simple(species = c("Acer rubrum", "Aspen tremuloides") ,
           country = c("Canada","Canada"),state_province = c("Ontario","Ontario"))


## End(Not run)
```

NSR_sources *Get information on sources used by the NSR*

Description

Return metadata about the current NSR sources

Usage

```
NSR_sources(...)
```

Arguments

...	Additional arguments passed to internal functions.

Value

Dataframe containing information about the sources used in the current NSR version.

Examples

```
{
sources <- NSR_sources()
}
```

NSR_template — *Make a template for an NSR query*

Description

NSR_template builds a template that can be populated to submit an NSR query.

Usage

```
NSR_template(nrow = 1)
```

Arguments

nrow	The number of rows to include in the template

Value

Template data.frame that can be populated and then used in NSR queries.

Examples

```
## Not run:

template<-NSR_template(nrow = 2)
template$genus<-"Acer"
template$species<-c("Acer rubrum", "Acer saccharum")
template$country<-"Canada"
template$user_id<-1:2
results <- NSR(occurrence_dataframe = template)


## End(Not run)
```

nsr_testfile — *Example NSR data*

Description

A sample dataset showing the proper formatting of NSR inputs.

Usage

```
nsr_testfile
```

Format

A data.frame with 22 observations of 5 variables:

taxon taxon, "Genus species" format, with specific epithet optional

country country that a species occurs within

state_province state/province that a species occurs within

county_parish county/parish that a species occurs within

user_id unique numeric ID that can be used to link to original dataset ...

Source

https://biendata.org

NSR_version *Get metadata on current NSR version*

Description

Return metadata about the current NSR version

Usage

```
NSR_version(...)
```

Arguments

...	Additional arguments passed to internal functions.

Value

Dataframe containing current NSR version number, build date, and code version.

Examples

```
{
NSR_version_metadata <- NSR_version()
}
```

Index

∗ **datasets**
nsr_testfile, 14

NSR, 2, *5–7*
NSR_citations, 3
NSR_data_dictionary, 4
NSR_local, 5, *8, 9*
NSR_local_build, 7
NSR_local_by_region, 8
NSR_local_remove, 9
NSR_local_status, 10
NSR_metadata, 10
NSR_political_divisions, 11
NSR_simple, 12
NSR_sources, 13
NSR_template, 14
nsr_testfile, 14
NSR_version, 15