Package: n2khab (via r-universe)

August 30, 2024

Title Providing Preprocessed Reference Data for Flemish Natura 2000 Habitat Analyses

Version 0.10.1

Description The n2khab package is an R package with preprocessing functions and standard reference data, useful for analyses regarding Flemish Natura 2000 habitats and regionally important biotopes (RIBs).

License GPL-3

URL https://inbo.github.io/n2khab, https://github.com/inbo/n2khab

BugReports https://github.com/inbo/n2khab/issues

Depends R (>= 3.5.0)

Imports assertthat, curl, dplyr, forcats, git2rdata (>= 0.4.0), magrittr, plyr, purrr, remotes, rlang, rprojroot, sf, stringr, tidyr (>= 1.0.0),

Encoding UTF-8

LazyData true

RoxygenNote 7.2.3

Suggests bib2df, digest, googledrive, jsonlite, knitr, mapview, n2khabmon, openssl, raster (>= 3.6-3), readxl, rmarkdown, terra, testthat (>= 3.0.0), tidyverse, tools, units, utils, withr

Remotes inbo/n2khabmon

VignetteBuilder knitr

Config/testthat/edition 3

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

RemoteUrl https://github.com/inbo/n2khab

RemoteRef HEAD

RemoteSha 2c2af9b4cd5e857b33b37afe0f6d47e9bad92c67

Contents

`checksum` *Calculate file checksums*

Description

The functions calculate the checksum (digest; hash value) of one or multiple files. They can be used to verify file integrity.

Usage

```
checksum(files, hash_fun = c("xxh64", "md5", "sha256"))

xxh64sum(files)

md5sum(files)

sha256sum(files)
```

Arguments

files	Character vector of file path(s). File path(s) can be absolute or relative.
hash_fun	String that defines the hash function. See *Usage* for allowed values; defaults to the first.

Details

A few cryptographic and non-cryptographic hash functions are implemented, either from the OpenSSL library (through openssl) or as embedded in the digest package.

Functions md5sum() etc. are simple shortcuts to checksum() with the appropriate hash function preset. Their names were chosen to match those of xxHash and GNU coreutils.

The cryptographic algorithms use the OpenSSL implementation and stream-hash the binary contents of the connections to the respective files. They turn the hash-format for binary streams by the openssl package into a regular hash string. Note that n2khab will mask tools::md5sum(), which is a standalone implementation.

Value

Named character vector with the same length as files and with the file names as names.

See Also

Other functions regarding file management for N2KHAB projects: download_zenodo(), fileman_folders(), fileman_up(), locate_n2khab_data()

Examples

```
# creating two different temporary files:
file1 <- tempfile()
file2 <- tempfile()
files <- c(file1, file2)
file.create(files)
con <- file(file2)
writeLines("some text", con)
close(con)

# computing alternative checksums:
checksum(files)
xxh64sum(files)
```

```
md5sum(files)
sha256sum(files)

## Not run:
# This will error:
files <- c(file1, file2, tempfile(), tempfile())
checksum(files)

## End(Not run)
```

`convert_base4frac_to_dec` *Convert a vector of base 4 fractions into (truncated) decimal integer values*

Description

Converts base 4 fractions, representing the full GRTS addresses from the raster data source `GRTSmaster_habitats` into decimal (i.e. base 10) integer values. Before the actual conversion happens, leading digits from the full GRTS address can be discarded according to the `level` specified by the user. Hence, the result may correspond to the GRTS ranking at a lower spatial resolution.

Usage

```
convert_base4frac_to_dec(x, level)
```

Arguments

`x`	A scalar or vector of base 4 fractions, originating from the `GRTSmaster_habitats` data source.
`level`	The number of leading digits to discard from the GRTS base 4 address, i.e. from the '`xxx...`' digits behind the decimal mark in the '`0.xxxxxxxxxxxxx`' base 4 fractions. Only values from `0` (maintain full address) to `12` (only return last digit) are sensible, as the GRTS addresses are `13` digits long.

Details

For example, the base 4 fraction `0.0000000000100` is converted into decimal integer `16` (= `4^2`) as long as the `level` argument is `10` or lower (if not, it will be `0`) and `0.0000000000101` is converted into either `17` (`level` <= `10`) or `1` (if `level` is `11` or `12`).

Long base 4 fractions seem to be handled and stored easier than long (base 4) integers. This approach follows the one of Stevens & Olsen (2004) to represent the reverse hierarchical order in a GRTS sample.

The function works on a vector and retains `NA` values. As such, it can be used in `raster::calc()`. When writing such a raster to a file, it is recommended to use the `INT4U` data type (see `dataType`).

Value

The corresponding decimal (i.e. base 10) integer scalar or vector.

References

Stevens D.L. & Olsen A.R. (2004). Spatially Balanced Sampling of Natural Resources. Journal of the American Statistical Association 99 (465): 262–278. doi:10.1198/016214504000000250.

See Also

Other functions involved in processing the 'GRTSmaster_habitats' data source: `convert_dec_to_base4frac()`, `read_GRTSmh_base4frac()`, `read_GRTSmh_diffres()`, `read_GRTSmh()`

Examples

```
oldoption <- options(list(digits = 15, scipen = 999))
# one scalar:
convert_base4frac_to_dec(0.1010101010101, level = 0)
# vector, level 0:
convert_base4frac_to_dec(c(NA, 0.1010101010101), level = 0)
# vector, level 5:
convert_base4frac_to_dec(c(NA, 0.1010101010101), level = 5)
# same vector, all sensible levels computed:
sapply(0:12, function(i) {
  convert_base4frac_to_dec(c(NA, 0.1010101010101),
    level = i
  )
})
options(oldoption)
```

`convert_dec_to_base4frac` *Convert a vector of values from the* `GRTSmaster_habitats` *data source to base 4 fractions*

Description

Converts decimal (i.e. base 10) integer values from the raster data source `GRTSmaster_habitats` into base 4 fractions, using a precision of 13 digits behind the decimal mark (as needed to cope with the range of values). For example, the integer `16` (`= 4^2`) is converted into `0.0000000000100` and `4^12` is converted into `0.1000000000000`.

Usage

```
convert_dec_to_base4frac(x)
```

Arguments

x	A decimal (i.e. base 10) scalar or vector of integer values from the GRTSmaster_habitats raster.

Details

Long base 4 fractions seem to be handled and stored easier than long (base 4) integers. This approach follows the one of Stevens & Olsen (2004) to represent the reverse hierarchical order in a GRTS sample as base-4-fraction addresses.

The function works on a vector and retains NA values. As such, it can be used in raster::calc(). When writing such a raster to a file, it is needed to use the FLT8S data type (see dataType). Otherwise several digits will change.

The function is based on code from the baseConvert() function in Will Gray's Gmisc package.

Value

The corresponding base4 scalar or vector, stored as a fraction.

References

Stevens D.L. & Olsen A.R. (2004). Spatially Balanced Sampling of Natural Resources. Journal of the American Statistical Association 99 (465): 262–278. doi:10.1198/016214504000000250.

See Also

Other functions involved in processing the 'GRTSmaster_habitats' data source: convert_base4frac_to_dec(), read_GRTSmh_base4frac(), read_GRTSmh_diffres(), read_GRTSmh()

Examples

```
oldoption <- options(list(digits = 15, scipen = 999))
convert_dec_to_base4frac(c(14, 15, NA, 456))
options(oldoption)
```

---

download_zenodo *Get data from a Zenodo archive*

Description

This function will download an entire archive from Zenodo (https://zenodo.org). It only works for Zenodo created DOI (not when the DOI is for example derived from Zookeys.)

Usage

```
download_zenodo(doi, path = ".", parallel = TRUE, quiet = FALSE)
```

Arguments

doi	a doi pointer to the Zenodo archive starting with '10.5281/zenodo.'. See examples.
path	Path where the data must be downloaded. Defaults to the working directory.
parallel	Logical. If TRUE (the default), files will be downloaded concurrently for multi-file records. Of course, the operation is limited by bandwidth and traffic limitations.
quiet	Logical (FALSE by default). Do you want to suppress informative messages (not warnings)?

Author(s)

Hans Van Calster, <hans.vancalster@inbo.be>

Floris Vanderhaeghe, <floris.vanderhaeghe@inbo.be>

See Also

Other functions regarding file management for N2KHAB projects: checksum(), fileman_folders(), fileman_up(), locate_n2khab_data()

Examples

```
## Not run:
# Example download of an archive containing a single zip
download_zenodo(doi = "10.5281/zenodo.1283345")
download_zenodo(doi = "10.5281/zenodo.1283345", quiet = TRUE)
# Example download of an archive containing multiple files
# using parallel download
# (multiple files will be simultaneously downloaded)
download_zenodo(doi = "10.5281/zenodo.1172801", parallel = TRUE)
# Example download of an archive containing a single pdf file
download_zenodo(doi = "10.5281/zenodo.168478")

## End(Not run)
```

---

env_pressures *Documentation of included data source 'env_pressures'*

Description

'env_pressures' is a data source in the vc-format which provides a checklist of environmental pressures, represented by codes, together with the pressure-class and the textual explanation (with optional remarks). The codes of environmental pressures, pressure-classes and explanations are explained in the data source namelist (which can accommodate multiple languages).

Format

A vc-formatted data source. As such, it corresponds to a data frame with 35 rows and 3 variables:

ep_code Code of the environmental pressure, as a factor. This is the ID for use in diverse workflows and datasets. Corresponding names and abbreviations in multiple languages are stored in `namelist` (as name and shortname, respectively). The abbreviation may be seen as an alternative, language-dependent code. Contains no duplicates!

ep_class A code explained by `namelist`, corresponding to the environmental pressure's class. Is a factor.

explanation A code explained by `namelist`, corresponding to the explanation on the environmental pressure, and optional remarks. Explanation and remarks are stored in `namelist` (as name and shortname, respectively)

Typical way of loading

```
read_env_pressures()
read_env_pressures(lang = "nl")
```

Corresponding datafiles in the installed package

```
textdata/env_pressures.csv
textdata/env_pressures.yml
```

Source

The latest worksheet of this googlesheet. Currently, the googlesheet and the data source are both kept up-to-date.

See Also

`read_env_pressures`

Other n2khab-referencelists: `namelist`, `types`

`expand_types` *Expand a 'type' column in a data frame*

Description

Takes a data frame with a column of type codes (*main type* or *subtype* codes), and, under certain conditions, adds new rows with codes of the associated *subtypes* and *main types*, respectively. It allows to do sensible selections and joins with interpreted forms of the `habitatmap_stdized` and `watersurfaces_hab` data sources: `habitatmap_terr`, `read_watersurfaces_hab(interpreted = TRUE)`. If the data frame has one or more grouping variables, by default the operation is done independently for each group in turn.

Usage

```
expand_types(x, type_var = "type", use_grouping = TRUE, strict = TRUE)
```

Arguments

x	An object of class data.frame.
type_var	A string. The name of the data frame variable that holds the type codes. Defaults to type.
use_grouping	Logical. If the data frame has one or more grouping variables (class grouped_df), is the operation to be performed independently for each group in turn?
strict	Logical. Apply conditions before expanding subtype codes to main type codes?

Details

The extra rows in the data frame take the values for other variables from the rows with which they are associated, based on the subtype - main type relation. Type codes in the data frame are verified to comply with the codes from the types data source. A warning is given when they don't.

Main type codes are always expanded with the subtype codes that belong to it.

The applied approach to add main type codes only makes sense assuming that the result is to be confronted with one of the *above listed* geospatial data sources.

In order to add main type codes based on subtype codes that are present in the type column, specific conditions have to be met:

- for 2330: both subtype codes must be present
- for 5130: 5130_hei must be present (note that only the main type code occurs in the targeted data sources)
- for 6230: 6230_ha, 6230_hmo and 6230_hnk must be present (not the rare 6230_hnk)
- for 91E0: 91E0_va, 91E0_vm and 91E0_vn must be present (not the rarer 91E0_sf, 91E0_vc and 91E0_vo)

However, it is possible to relax this requirement by setting strict = FALSE. This will add the main type code whenever *one* corresponding subtype code is present. In all cases no other main type codes are added apart from 2330, 5130, 6230 and 91E0. This is because the data sources with which the result is to be matched (see Description) don't contain certain main type codes, and because it makes no sense in other cases (rbbkam, rbbvos, rbbzil & 9120 in the habitatmap do not refer to a main type but to an non-defined subtype with no specific code).

Value

A data frame, either identical or longer than the input data frame.

See Also

read_types, read_habitatmap_terr, read_watersurfaces_hab

Examples

```
library(dplyr)
x <-
  n2khabmon::read_scheme_types() %>%
  filter(scheme == "GW_05.1_terr")
expand_types(x)
expand_types(x, strict = FALSE)

x <-
  n2khabmon::read_scheme_types() %>%
  filter(scheme == "GW_05.1_terr") %>%
  group_by(typegroup)
expand_types(x)
expand_types(x, use_grouping = FALSE) # equals above example

x <-
  tribble(
    ~mycode, ~obs,
    "2130", 5,
    "2190", 45,
    "2330_bu", 8,
    "2330_dw", 8,
    "5130_hei", 7,
    "6410_mo", 78,
    "6410_ve", 4,
    "91E0_vn", 10
  )
expand_types(x, type_var = "mycode")
expand_types(x, type_var = "mycode", strict = FALSE)
```

---

fileman_folders *Create a standard data folder structure and return the path to the* n2khab_data *folder*

Description

This function will check for the existence of default data folders, create them if necessary, and return the path to the n2khab_data folder.

Usage

```
fileman_folders(root = c("rproj", "git"), path = NA)
```

Arguments

root	Character string indicating whether the root folder of the current git repository or the root folder of the current Rstudio project should be used as the folder where you want the data folder structure to be created. Can be "rproj" (the default) for an RStudio R project or "git" for a git repository.

path — An optional argument to specify a custom path to a folder where you want the data folder structure to be created. Default is NA (no custom path).

Details

In n2khab projects a standardized folder setup is used for binary data, as explained in the vignette on data storage (run `vignette("v020_datastorage")`). The functions creates the folders `n2khab_data`, `n2khab_data/10_raw` and `n2khab_data/20_processed`, or prints a message if these already exist. The function returns the path to `n2khab_data`.

Value

A character string that gives the absolute path to the `n2khab_data/` folder.

See Also

Other functions regarding file management for N2KHAB projects: `checksum()`, `download_zenodo()`, `fileman_up()`, `locate_n2khab_data()`

Examples

```
## Not run:
fileman_folders()
datapath <- fileman_folders(root = "git")

## End(Not run)
```

---

fileman_up *Climb up in the file system hierarchy to find a file or folder*

Description

Searches for a specific file or folder, starting from the `start` directory and sequentially climbing up one directory level at a time. The first match causes this sequence to stop and the full path will be returned.

Usage

```
fileman_up(name, start = ".", levels = 10)
```

Arguments

name — Name of file or folder to search for. An exact match is needed. The matching is case sensitive.

start — String. Directory to start searching from.

levels — Integer. How many levels to sequentially climb up in the file hierarchy, if the file or folder is not found in the `start` directory?

Details

Symbolic links are matched, and in the returned path they are converted.

Value

The path to the specified folder or file (string), or NULL if not found.

See Also

Other functions regarding file management for N2KHAB projects: checksum(), download_zenodo(), fileman_folders(), locate_n2khab_data()

Examples

```
## Not run:
fileman_up("n2khab_data")

## End(Not run)
```

---

locate_n2khab_data *Locate the* n2khab_data *directory*

Description

Returns the absolute path of the n2khab_data directory by searching upwards in the directory hierarchy (starting in the working directory); the first hit will be returned.

Usage

```
locate_n2khab_data()
```

Details

If the n2khab_data_path option (or environment variable N2KHAB_DATA_PATH) is set, that value will be returned instead (see n2khab_options()).

See the data management advice in the vignette on data storage (run vignette("v020_datastorage")) for more information.

See Also

Other functions regarding file management for N2KHAB projects: checksum(), download_zenodo(), fileman_folders(), fileman_up()

Examples

```
## Not run:
locate_n2khab_data()

## End(Not run)
```

`n2khab_options` *Package configuration through options and environment variables*

Description

The package can be configured by means of options or environment variables. These will influence the behaviour of certain functions. Each option has its sibling environment variable. When both have a value, the *option* will be given priority.

This function queries these options and environment variables, and returns the resulting state for each of them (not distinguishing between options or environment variables as the source).

Usage

```
n2khab_options()
```

Details

Options are typically harder to isolate from the R code that you collaborate on and share through a repository. This is especially the case when using `renv`: it requires `.Rprofile` as part of your project in the working directory, which prevents `.Rprofile` files elsewhere on the system from being used.

Consequently, it is advised to:

- use `options()` where this affects behaviour that must be the same across users and machines for reproducibility. Put these inside your script, or at least in an `.Rprofile` file that is shared together with the other project files. Example: which package to use to represent raster objects.
- use environment variables where behaviour must be machine-specific, e.g. to override the default location of the `n2khab_data` directory (can also be needed when using `reprex::reprex()`). For example, you can create an `.Renviron` file in your working directory and ignore it in distributed version control. Or you can set the environment variable at a higher level, e.g. in an `.Renviron` file in your home directory. See base::Startup for more information.

Value

A data frame with the names and values of possible options. Missing values are returned as `NA`.

Description of options and environment variables

option	environment variable	type	description
n2khab_data_path	N2KHAB_DATA_PATH	string	Path of the n2khab_data directory. Takes priority over the default
n2khab_use_raster	N2KHAB_USE_RASTER	logical	Should the raster package be used to return raster objects? The t

Examples

```
n2khab_options()

oldopt <- options(n2khab_use_raster = TRUE)
n2khab_options()
options(oldopt)

# Unacceptable values yield an error message;
# the data frame is still returned with NA:
oldopt <- options(n2khab_data_path = 0, n2khab_use_raster = TRUE)
n2khab_options()
options(oldopt)
```

---

namelist *Documentation of included data source 'namelist'*

Description

'namelist' is a data source in the vc-format which provides names and (optionally) shortnames for IDs/codes used in other data sources. Multiple languages are supported.

Format

A vc-formatted data source. As such, it corresponds to a data frame with many rows and 4 variables:

code A code used elsewhere.

lang An IETF BCP 47 language tag, such as `"en"` or `"nl"`, to identify the language of `name` and `shortname`.

name The name corresponding to `code` and `lang`.

shortname Optionally, a shorter variant of `name`.

Typical way of loading

```
read_namelist()
read_namelist(lang = "nl")
```

Corresponding datafiles in the installed package

`textdata/namelist.tsv`

`textdata/namelist.yml`

Source

The 'namelist' data source has got its contents from the sources of several other n2khab-referencelists (see the source of those).

See Also

`read_namelist`

Other n2khab-referencelists: `env_pressures`, `types`

`read_admin_areas` *Return one of the available geospatial data sources representing administrative areas*

Description

Returns an administrative geospatial data source. The coordinate reference system is 'BD72 / Belgian Lambert 72' (EPSG-code 31370).

Usage

```
read_admin_areas(
 file = file.path(locate_n2khab_data(), c("10_raw/flanders", "10_raw/provinces",
    "10_raw/sac")),
  dsn = c("flanders", "provinces", "sac")
)
```

Arguments

`file`	The absolute or relative file path of the data source. The default follows the data management advice in the vignette on data storage (run `vignette("v020_datastorage")`). It uses the first `n2khab_data` folder that is found when sequentially climbing up 0 to 10 levels in the file system hierarchy, starting from the working directory.
`dsn`	A string, conforming to one of the data source names listed under *Usage*. Considering the default values of the `path` and `file` arguments, it is recommended to just use this argument. Defaults to `"flanders"`. Different data sources are handled differently by the function. The `dsn` argument states which data source is requested. It is also used to determine `file` if that argument is not specified, in order to select the right data file(s) from the `n2khab_data` folder.

Details

See section *Usage* to see which N2KHAB data sources are available. You get a list of the data source names and related information with `XXXXXXXXX`. You are referred to the raw N2KHAB-data collection at Zenodo to learn more about these data sources.

Value

A Simple feature collection of geometry type `MULTIPOLYGON` or `POLYGON`.

Examples

```
## Not run:
flanders <- read_admin_areas(dsn = "flanders")
provinces <- read_admin_areas(dsn = "provinces")
sac <- read_admin_areas(dsn = "sac")

## End(Not run)
```

`read_ecoregions` *Return the* `ecoregions` *data source as an* `sf` *object*

Description

Returns the raw data source `ecoregions`, with unique polygon identifiers `polygon_code` and `polygon_id`. Multiple polygons can have the same `region_name`. The coordinate reference system is 'BD72 / Belgian Lambert 72' (EPSG-code 31370).

Usage

```
read_ecoregions(file = file.path(locate_n2khab_data(), "10_raw/ecoregions"))
```

Arguments

`file`	The absolute or relative file path of the data source. The default follows the data management advice in the vignette on data storage (run `vignette("v020_datastorage")`). It uses the first `n2khab_data` folder that is found when sequentially climbing up 0 to 10 levels in the file system hierarchy, starting from the working directory.

Details

Original columns of the raw data source were mapped as:

- `CODE` -> `polygon_code`
- `NR` -> `polygon_id`
- `REGIO` -> `region_name`
- `DISTRICT` -> `district_name`

Apart from the label, there is no complementary information between `polygon_code` and `polygon_id`.

Value

A Simple feature collection of geometry type MULTIPOLYGON.

Examples

```
## Not run:
ecoregions <- read_ecoregions()
ecoregions

## End(Not run)
```

---

read_env_pressures *Return the 'env_pressures' data source as a tibble with human-readable attributes*

Description

Returns the included data source env_pressures as a tibble. Names, shortnames, explanations and optional remarks from namelist are added, in English by default.

Usage

```
read_env_pressures(
  path = pkgdatasource_path("textdata/env_pressures", ".yml"),
  file = "env_pressures",
  file_namelist = "namelist",
  lang = "en"
)
```

Arguments

path	Location of the data sources env_pressures and namelist. The default is to use the location of the data sources as delivered by the installed package.
file	The filename of the env_pressures data source, without extension. The default is to use the file delivered by the installed package.
file_namelist	The filename of the namelist data source, without extension. The default is to use the file delivered by the installed package.
lang	An IETF BCP 47 language tag, such as "en" or "nl", to specify the language of human-readable attributes to be returned in the tibble.

Details

env_pressures is a data source in the vc-format which provides a checklist of environmental pressures, represented by codes, together with the pressure-class and the textual explanation.

read_env_pressures() reads the env_pressures data source, adds human-readable attributes and returns it as a tibble. A tibble is a data frame that makes working in the tidyverse a little easier. By default, the data version delivered with the package is used and English text (lang = "en") is returned for names of environmental pressures and pressure-classes, and for textual explanations and remarks.

Value

The env_pressures data frame as a tibble, with human-readable text added for environmental pressures, pressure-classes and textual explanations and remarks according to the lang argument. The tibble has 35 rows and 7 variables. See env_pressures for documentation of the data-source's contents. See namelist for the link between codes or other identifiers and the corresponding text.

The human-readable attributes are represented by the following variables:

ep_abbrev A (language-dependent) abbreviation (alternative code) of the environmental pressure. Is a factor with the level order coinciding with that of ep_code.

ep_name The name of the environmental pressure. Is a factor with the level order coinciding with that of ep_code.

ep_class_name The name of the environmental pressure's class. Is a factor with the level order coinciding with that of ep_class.

explanation An explanation of the environmental pressure.

remarks Optional remarks about the environmental pressure.

Recommended usage

```
read_env_pressures()

read_env_pressures(lang = "nl")
```

See Also

env_pressures

Other reading functions for n2khab-referencelists: read_namelist(), read_types()

Examples

```
read_env_pressures()
read_env_pressures(lang = "nl")
```

`read_GRTSmh` *Return the* `GRTSmaster_habitats` *data source or a 10-layered variant as a SpatRaster object*

Description

By default, the `GRTSmaster_habitats` data source is returned as a single-layered SpatRaster object with decimal integer ranking numbers as values. If `brick = TRUE`, a ten-layered SpatRaster is returned (data source `GRTSmh_brick`; resolution 32 m) with the decimal integer ranking numbers of 10 hierarchical levels of the GRTS cell addresses, including the one from `GRTSmaster_habitats` (with GRTS cell addresses at the resolution level). The coordinate reference system is 'BD72 / Belgian Lambert 72' (EPSG-code 31370).

Usage

```
read_GRTSmh(
  file = file.path(locate_n2khab_data(),
    c("10_raw/GRTSmaster_habitats/GRTSmaster_habitats.tif",
    "20_processed/GRTSmh_brick/GRTSmh_brick.tif")),
  brick = FALSE
)
```

Arguments

`file`	The absolute or relative file path of the data source. The default follows the data management advice in the vignette on data storage (run `vignette("v020_datastorage")`). It uses the first `n2khab_data` folder that is found when sequentially climbing up 0 to 10 levels in the file system hierarchy, starting from the working directory.
`brick`	Logical; determines whether the single- or ten-layered SpatRaster is returned. See the Details section.

Details

The data source `GRTSmaster_habitats`, provided and documented in Zenodo, is a monolayered GeoTIFF file covering the whole of Flanders and the Brussels Capital Region at a resolution of 32 m. Its values are unique decimal integer ranking numbers from the GRTS algorithm applied to the Flemish and Brussels area. Beware that not all GRTS ranking numbers are present in the data source, as the original GRTS raster has been clipped with the Flemish outer borders (i.e., not excluding the Brussels Capital Region).

The GRTS algorithm uses a quadrant-recursive, hierarchically randomized function that maps the unit square to the unit interval, resulting in a base-4 GRTS address for each location (see `read_GRTSmh_base4frac`). The ranking numbers in `GRTSmaster_habitats` are base-10 numbers and follow the reverse hierarchical order: each consecutive subset of ranking numbers corresponds to a spatially balanced sample of locations. Hence, it allows dynamical sample sizes. More information on the GRTS algorithm can be found in Stevens & Olsen (2003, 2004) and in the GRTS and spsurvey packages.

Depending on the value of the `brick` argument, the function either returns the `GRTSmaster_habitats` data source as a single-layered SpatRaster (`brick = FALSE`), or (`brick = TRUE`) returns the `GRTSmh_brick`

data source as a ten-layered SpatRaster (resolution 32 m) with the decimal integer ranking numbers of 10 hierarchical levels of the GRTS cell addresses, including the one from `GRTSmaster_habitats` (with GRTS cell addresses at the resolution level). The `GRTSmh_brick` data source is a processed dataset (ten-layered GeoTIFF), available at Zenodo, and can only be returned by the function when it is already present as a file. See R-code in the n2khab-preprocessing repository for its creation from the `GRTSmaster_habitats` data source.

Both GeoTIFFs (`GRTSmaster_habitats`, `GRTSmh_brick`) use the `INT4S` datatype.

The higher-level ranking numbers of the ten-layered variant allow spatially balanced samples at lower spatial resolution than that of 32 m, and can also be used for aggregation purposes. The provided hierarchical levels correspond to the resolutions vector `32 * 2^(0:9)` (minimum: 32 meters, maximum: 16384 meters), with the corresponding SpatRaster layers named as `level0` to `level9`.

Value

A single- or a ten-layered SpatRaster object, always with 21041043 cells.

If the package is configured to use the raster package (see `n2khab_options()`), a RasterLayer is returned if `brick = FALSE` and a RasterBrick if `brick = TRUE`.

References

Stevens D.L. & Olsen A.R. (2003). Variance estimation for spatially balanced samples of environmental resources. Environmetrics 14 (6): 593–610. doi:10.1002/env.606.

Stevens D.L. & Olsen A.R. (2004). Spatially Balanced Sampling of Natural Resources. Journal of the American Statistical Association 99 (465): 262–278. doi:10.1198/016214504000000250.

See Also

Other functions involved in processing the 'GRTSmaster_habitats' data source: `convert_base4frac_to_dec()`, `convert_dec_to_base4frac()`, `read_GRTSmh_base4frac()`, `read_GRTSmh_diffres()`

Examples

```
## Not run:
# This example supposes that your working directory or a directory up to 10
# levels above has
# the 'n2khab_data' folder AND that the latest version of the
# 'GRTSmaster_habitats' data source is present in the default subdirectory.
# In all other cases, this example won't work but at least you can consider
# what to do.
r <- read_GRTSmh()
r
r10 <- read_GRTSmh(brick = TRUE)
r10

## End(Not run)
```

`read_GRTSmh_base4frac` *Return the processed data source* `GRTSmh_base4frac` *as a SpatRaster*

Description

The `GRTSmh_base4frac` data source is like a mirror to `GRTSmaster_habitats`, holding the ranking numbers as base 4 fractions. The function returns it as a SpatRaster in the Belgian Lambert 72 CRS (EPSG-code 31370).

Usage

```
read_GRTSmh_base4frac(
  file = file.path(locate_n2khab_data(),
    "20_processed/GRTSmh_base4frac/GRTSmh_base4frac.tif")
)
```

Arguments

`file`	The absolute or relative file path of the data source. The default follows the data management advice in the vignette on data storage (run `vignette("v020_datastorage")`). It uses the first `n2khab_data` folder that is found when sequentially climbing up 0 to 10 levels in the file system hierarchy, starting from the working directory.

Details

The data source file, read by the function, is a monolayered GeoTIFF in the `FLT8S` datatype and is available at Zenodo. In `GRTSmh_base4frac`, the decimal (i.e. base 10) integer values from the raster data source `GRTSmaster_habitats` (see `read_GRTSmh`) have been converted into base 4 fractions, using a precision of 13 digits behind the decimal mark (as needed to cope with the range of values). For example, the integer `16` (= `4^2`) has been converted into `0.0000000000100` and `4^12` has been converted into `0.1000000000000`.

Long base 4 fractions seem to be handled and stored easier than long (base 4) integers. This approach follows the one of Stevens & Olsen (2004) to represent the reverse hierarchical order in a GRTS sample as base-4-fraction addresses.

See R-code in the n2khab-preprocessing repository for the creation from the `GRTSmaster_habitats` data source.

Beware that not all GRTS ranking numbers are present in the data source, as the original GRTS raster has been clipped with the Flemish outer borders (i.e., not excluding the Brussels Capital Region).

Also, be warned that R does not regard the values as base 4, but as base 10. So, what really matters is only the notation with many digits, to be *regarded* as a base 4 fraction.

Value

A SpatRaster with 21041043 cells.

If the package is configured to use the raster package (see `n2khab_options()`), a RasterLayer is returned instead.

References

Stevens D.L. & Olsen A.R. (2004). Spatially Balanced Sampling of Natural Resources. Journal of the American Statistical Association 99 (465): 262–278. doi:10.1198/016214504000000250.

See Also

Other functions involved in processing the 'GRTSmaster_habitats' data source: `convert_base4frac_to_dec()`, `convert_dec_to_base4frac()`, `read_GRTSmh_diffres()`, `read_GRTSmh()`

Examples

```
## Not run:
# This example supposes that your working directory or a directory up to 10
# levels above has
# the 'n2khab_data' folder AND that the latest version of the
# 'GRTSmh_base4frac' data source is present in the default subdirectory.
# In all other cases, this example won't work but at least you can consider
# what to do.
oldopt <- options(scipen = 999, digits = 15)
r <- read_GRTSmh_base4frac()
r
options(oldopt)

## End(Not run)
```

`read_GRTSmh_diffres` *Return a SpatRaster or an* `sf` *polygon layer from the processed data source* `GRTSmh_diffres`

Description

The `GRTSmh_diffres` data source is derived from `GRTSmh_brick`. It provides the hierarchical levels 1 to 9 of the GRTS cell addresses at the corresponding spatial resolution. The function returns one selected level, either as a SpatRaster or as an `sf` polygon layer (in the latter case, only levels 4 to 9 are provided). The coordinate reference system is 'BD72 / Belgian Lambert 72' (EPSG-code 31370).

Usage

```
read_GRTSmh_diffres(
  dir = file.path(locate_n2khab_data(), "20_processed/GRTSmh_diffres"),
  level,
  polygon = FALSE
)
```

Arguments

dir	The data source directory (absolute or relative). The default follows the data management advice in the vignette on data storage (run vignette("v020_datastorage")). It uses the first n2khab_data folder that is found when sequentially climbing up 0 to 10 levels in the file system hierarchy, starting from the working directory.
level	Integer in the range from 1 to 9; determines the spatial resolution. See the Details section.
polygon	Logical; determines whether a polygon layer or a SpatRaster is returned. See the Details section.

Details

The GRTSmh_diffres data source file is a file collection (available at Zenodo), composed of nine monolayered GeoTIFF files of the INT4S datatype plus a GeoPackage with six polygon layers:

- The polygon layers in the GeoPackage are the dissolved, polygonized versions of levels 4 to 9 of the GRTSmh_brick data source (see read_GRTSmh). This means that they provide the decimal (i.e. base 10) integer values of these *higher hierarchical levels* of the GRTS cell addresses of the raw data source GRTSmaster_habitats. Hence, the polygons are typically squares that correspond to the GRTS cell at the specified hierarchical level. The polygon layer is however restricted to the non-NA cells of the original GRTSmaster_habitats raster. Consequently, a part of the polygons is clipped along the Flemish border. Levels 1 to 3 are not provided for the whole of Flanders, because this would inflate the GPKG file. You can look at the source code to do such things.
- The GeoTIFF files provide the respective levels 1 to 9 of the GRTSmh_brick data source in a raster format, at the resolution that corresponds to the GRTS cell at the specified hierarchical level. The presence of NA cells around Flanders at level 0 implies that, with decreasing resolution, the raster's extent increases and larger areas outside Flanders are covered by non-NA cells along the border.

The function returns the selected level either as an sf polygon layer or as a SpatRaster, depending on the polygon argument.

The higher-level ranking numbers (compared to the original level 0) allow spatially balanced samples at lower spatial resolution than that of 32 m, and can also be used for aggregation purposes.

The levels 1 to 9 correspond to the resolutions vector 32 * 2^(1:9) in meters:

level	resolution (m)
1	64
2	128
3	256
4	512
5	1024
6	2048
7	4096
8	8192
9	16384

See R-code in the n2khab-preprocessing repository for the creation from the GRTSmh_brick data source.

Beware that not all GRTS ranking numbers at the specified level are provided, as the original GRTS raster has been clipped with the Flemish outer borders (i.e., not excluding the Brussels Capital Region).

Value

Either a SpatRaster or a Simple feature collection of geometry type POLYGON.

If the package is configured to use the raster package (see n2khab_options()), a RasterLayer is returned instead of a SpatRaster.

See Also

Other functions involved in processing the 'GRTSmaster_habitats' data source: convert_base4frac_to_dec(), convert_dec_to_base4frac(), read_GRTSmh_base4frac(), read_GRTSmh()

Examples

```
## Not run:
# This example supposes that your working directory or a directory up to 10
# levels above has
# the 'n2khab_data' folder AND that the latest version of the
# 'GRTSmh_diffres' data source is present in the default subdirectory.
# In all other cases, this example won't work but at least you can consider
# what to do.
r <- read_GRTSmh_diffres(level = 7)
r
p <- read_GRTSmh_diffres(level = 7, polygon = TRUE)
p

## End(Not run)
```

---

read_habitatmap *Return the data source* habitatmap *as an* sf *multipolygon layer*

Description

Returns the raw data source habitatmap (De Saeger et al., 2020) as a standardized sf multipolygon layer (tidyverse-styled, internationalized) in the Belgian Lambert 72 CRS (EPSG-code 31370). Given the size of the data source, this function takes a bit longer than usual to run.

Usage

```
read_habitatmap(
  file = file.path(locate_n2khab_data(), "10_raw/habitatmap"),
  filter_hab = FALSE,
  version = c("habitatmap_2020", "habitatmap_2018")
)
```

Arguments

file	The absolute or relative file path of the data source. The default follows the data management advice in the vignette on data storage (run vignette("v020_datastorage")). It uses the first n2khab_data folder that is found when sequentially climbing up 0 to 10 levels in the file system hierarchy, starting from the working directory.
filter_hab	If TRUE only polygons that (partially) contain habitat or a regionally important biotope (RIB) are returned. The default value is FALSE. This requires the corresponding version of the processed data source habitatmap_stdized to be present in its default location inside the n2khab_data folder.
version	Version ID of the data source. Defaults to the latest available version defined by the package.

Value

A Simple feature collection of type MULTIPOLYGON.

References

- De Saeger, S., Guelinckx, R., Oosterlynck, P., De Bruyn, A., Debusschere, K., Dhaluin, P., Erens, R., Hendrickx, P., Hennebel, D., Jacobs, I., Kumpen, M., Opdebeeck, J., Spanhove, T., Tamsyn, W., Van Oost, F., Van Dam, G., Van Hove, M., Wils, C., Paelinckx, D. (2020). Biologische Waarderingskaart en Natura 2000 Habitatkaart, uitgave 2020. (Rapporten van het Instituut voor Natuur- en Bosonderzoek; Nr. 35). Instituut voor Natuur- en Bosonderzoek (INBO). doi:10.21436/inbor.18840851.
- De Saeger, S., Oosterlynck, P. & Paelinckx, D. (2017). The Biological Valuation Map (BVM): a field-driven survey of land cover and vegetation in the Flemish Region of Belgium. Documents phytosociologiques - Actes du colloque de Saint-Mandé 2012 - Prodrome et cartographie des végétations de France - 2017. Vol. 6: 372-382.

See Also

Other functions involved in processing the 'habitatmap' data source: read_habitatmap_stdized(), read_habitatmap_terr(), read_watersurfaces_hab()

Examples

```
## Not run:
# This example supposes that your working directory or a directory up to 10
# levels above has the 'n2khab_data' folder AND that the latest version of
# the 'habitatmap'
# data source is present in the default subdirectory.
# In all other cases, this example won't work but at least you can
# consider what to do.

hm <- read_habitatmap()
hm

## End(Not run)
```

`read_habitatmap_stdized` *Return the data source* `habitatmap_stdized` *as a list of two objects*

Description

`read_habitatmap_stdized` returns the data source `habitatmap_stdized` as a list of two objects:

- `habitatmap_polygons`: an sf object in the Belgian Lambert 72 CRS (EPSG-code 31370) with all polygons of the `habitatmap` that contain habitat or a regionally important biotope (RIB).
- `habitatmap_types`: a tibble with information on the habitat and RIB types (HAB1, HAB2,..., HAB5) that occur within each polygon of `habitatmap_polygons`.

Usage

```
read_habitatmap_stdized(
  file = file.path(locate_n2khab_data(),
    "20_processed/habitatmap_stdized/habitatmap_stdized.gpkg"),
  version = c("habitatmap_stdized_2020_v1", "habitatmap_stdized_2018_v2",
    "habitatmap_stdized_2018_v1")
)
```

Arguments

`file`	The absolute or relative file path of the data source. The default follows the data management advice in the vignette on data storage (run `vignette("v020_datastorage")`). It uses the first `n2khab_data` folder that is found when sequentially climbing up 0 to 10 levels in the file system hierarchy, starting from the working directory.
`version`	Version ID of the data source. Defaults to the latest available version defined by the package.

Details

The data source `habitatmap_stdized` is the processed version of the raw data source `habitatmap` (De Saeger et al., 2020). Every polygon in the `habitatmap` can consist of maximum 5 different types. This information is stored in the columns 'HAB1', HAB2',..., 'HAB5' of the attribute table. The fraction of each type within the polygons is stored in the columns 'PHAB1', 'PHAB2', ..., 'PHAB5'.

The data source `habitatmap_stdized` is a GeoPackage, available at Zenodo, that contains:

- `habitatmap_polygons`: a spatial layer with every `habitatmap` polygon that contains a habitat or RIB type listed in `types`.
- `habitatmap_types`: a table with the types that occur in each polygon.

The processing of the `habitatmap_types` tibble included following steps:

- For some polygons the type is uncertain, and the type code in the raw habitatmap data source consists of 2 or 3 possible types, separated with a ',' . The different possible types are split up and one row is created for each of them, with phab for each new row simply set to the original value of phab. The variable certain will be FALSE if the original code consists of 2 or 3 possible types, and TRUE if only one type is provided.
- Some polygons contain both a standing water habitat type and rbbmr: 3130_rbbmr, 3140_rbbmr, 3150_rbbmr and 3160_rbbmr. Since habitatmap_stdized_2020_v1, the two types 31xx and rbbmr are split up and one row is created for each of them, with phab for each new row simply set to the original value of phab. The variable certain in this case will be TRUE for both types.
- After those first two steps, a given polygon could contain the same type with the same value for certain repeated several times, e.g. when 31xx_rbbmr is present with phab = yy% and 31xx is present with phab = zz%. In that case the rows with the same polygon_id, type and certain were gathered into one row and the respective phab values were added up.
- For some polygons the original type code in the habitatmap was not consistent with general coding syntax or with the type codes from the types data source. In that case the code was adjusted.

The R-code for creating the habitatmap_stdized data source can be found in the n2khab-preprocessing repository.

Value

A list of two objects:

- habitatmap_polygons: an sf object of habitatmap polygons with two attribute variables
  - polygon_id
  - description_orig: polygon description based on the orginal type codes in the raw habitatmap
- habitatmap_types: a tibble with following variables
  - polygon_id
  - type: habitat or RIB type listed in types.
  - certain: TRUE when the type is certain and FALSE when the type is uncertain.
  - code_orig: original type code in raw habitatmap.
  - phab: proportion of polygon covered by type, as a percentage.

  Since version habitatmap_stdized_2020_v1, rows are unique only by the combination of the polygon_id, type and certain columns.

References

- De Saeger, S., Guelinckx, R., Oosterlynck, P., De Bruyn, A., Debusschere, K., Dhaluin, P., Erens, R., Hendrickx, P., Hennebel, D., Jacobs, I., Kumpen, M., Opdebeeck, J., Spanhove, T., Tamsyn, W., Van Oost, F., Van Dam, G., Van Hove, M., Wils, C., Paelinckx, D. (2020). Biologische Waarderingskaart en Natura 2000 Habitatkaart, uitgave 2020. (Rapporten van het Instituut voor Natuur- en Bosonderzoek; Nr. 35). Instituut voor Natuur- en Bosonderzoek (INBO). doi:10.21436/inbor.18840851.

- De Saeger, S., Oosterlynck, P. & Paelinckx, D. (2017). The Biological Valuation Map (BVM): a field-driven survey of land cover and vegetation in the Flemish Region of Belgium. Documents phytosociologiques - Actes du colloque de Saint-Mandé 2012 - Prodrome et cartographie des végétations de France - 2017. Vol. 6: 372-382.

See Also

Other functions involved in processing the 'habitatmap' data source: `read_habitatmap_terr()`, `read_habitatmap()`, `read_watersurfaces_hab()`

Examples

```
## Not run:
# This example supposes that your working directory or a directory up to 10
# levels above has the 'n2khab_data' folder AND that the latest version of
# the 'habitatmap_stdized'
# data source is present in the default subdirectory.
# In all other cases, this example won't work but at least you can
# consider what to do.

hms <- read_habitatmap_stdized()
hms_polygons <- hms$habitatmap_polygons
hms_types <- hms$habitatmap_types
hms_polygons
hms_types

## End(Not run)
```

---

read_habitatmap_terr *Return the data source* habitatmap_terr *as a list of two objects*

Description

`read_habitatmap_terr()` returns the data source `habitatmap_terr` as a list of two objects: `habitatmap_terr_polygons`, having the Belgian Lambert 72 CRS (EPSG-code 31370), and `habitatmap_terr_types`. `habitatmap_terr` is the further interpreted, terrestrial part of `habitatmap_stdized` (see `read_habitatmap_stdized`), which, in turn, is derived from the raw data source `habitatmap` (De Saeger et al., 2020). By default, occurrences of type `7220` are dropped because a more reliable data source is available for this habitat type (see `drop_7220` argument). Note: a type is a habitat (sub)type or a regionally important biotope (RIB).

Usage

```
read_habitatmap_terr(
  file = file.path(locate_n2khab_data(),
    "20_processed/habitatmap_terr/habitatmap_terr.gpkg"),
  keep_aq_types = TRUE,
  drop_7220 = TRUE,
```

```
  version = c("habitatmap_terr_2020_v1", "habitatmap_terr_2018_v2",
    "habitatmap_terr_2018_v1")
)
```

Arguments

file	The absolute or relative file path of the data source. The default follows the data management advice in the vignette on data storage (run vignette("v020_datastorage")). It uses the first n2khab_data folder that is found when sequentially climbing up 0 to 10 levels in the file system hierarchy, starting from the working directory.
keep_aq_types	Logical; TRUE by default. The data source habitatmap_terr aims at delineating all polygons with at least one (semi-)terrestrial type. For those polygons, it returns all known habitat types and RIBs as types. Hence, in several cases polygons do occur with a combination of terrestrial and aquatic types (see *Details* for a definition of 'aquatic'). Setting keep_aq_types = FALSE is convenient for use cases where one only wants to look at the (semi-)terrestrial types: this setting will discard all aquatic types in 'mixed' aquatic/terrestrial polygons. As each polygon always has at least one (semi-)terrestrial type, this will not affect the number of polygons returned, only the number of types.
drop_7220	Logical; TRUE by default. Should occurrences of type 7220 be dropped from the result? To get more accurate information about type 7220, notably its occurrences, surface area and other characteristics, it is advised to use the habitatsprings data source and not habitatmap_terr - see read_habitatsprings().
version	Version ID of the data source. Defaults to the latest available version defined by the package.

Details

habitatmap_terr was derived from habitatmap_stdized as follows:

- it excludes all polygons that are most probably aquatic habitat or RIB. These are the polygons for which **all** habitat or RIB types are aquatic. In the process, a distinction was also made between 2190_a and 2190_overig. There is no exclusion of aquatic types when these coexist with terrestrial types in the same polygon. The aquatic types are the types for which hydr_class == "HC3" in the types data source (hydr_class is the hydrological class; cf. the output of read_types());
- it excludes types which most probably are *no* habitat or RIB at all. Those are the types where code_orig contains "bos" or is equal to "6510,gh" or "9120,gh";
- it translates several main type codes into a corresponding subtype which they almost always represent: 6410 -> 6410_mo, 6430 -> 6430_hf, 6510 -> 6510_hu, 7140 -> 7140_meso, 9130 -> 9130_end;
- it distinguishes types rbbhfl and rbbhf.

The data source habitatmap_terr is a GeoPackage, available at Zenodo, that contains:

- habitatmap_terr_polygons: a spatial polygon layer
- habitatmap_terr_types: a table with the types that occur in each polygon.

The R-code for creating the habitatmap_terr data source can be found in the n2khab-preprocessing repository.

Value

A list of two objects:

- habitatmap_terr_polygons: a Simple feature collection of geometry type MULTIPOLYGON with four attribute variables:
  - polygon_id
  - description_orig: polygon description based on the original type codes in the habitatmap data source
  - description: based on description_orig but with the interpreted type codes
  - source: states where description comes from: either habitatmap_stdized or habitatmap_stdized + interpretation
- habitatmap_terr_types: a tibble with the following variables (the first 4 being identical to those in habitatmap_stdized):
  - polygon_id
  - type: the interpreted habitat or RIB type
  - certain
  - code_orig
  - phab
  - source: states where type comes from: either habitatmap_stdized or habitatmap_stdized + interpretation

  Since version habitatmap_terr_2020_v1, rows are unique only by the combination of the polygon_id, type and certain columns.

References

- De Saeger, S., Guelinckx, R., Oosterlynck, P., De Bruyn, A., Debusschere, K., Dhaluin, P., Erens, R., Hendrickx, P., Hennebel, D., Jacobs, I., Kumpen, M., Opdebeeck, J., Spanhove, T., Tamsyn, W., Van Oost, F., Van Dam, G., Van Hove, M., Wils, C., Paelinckx, D. (2020). Biologische Waarderingskaart en Natura 2000 Habitatkaart, uitgave 2020. (Rapporten van het Instituut voor Natuur- en Bosonderzoek; Nr. 35). Instituut voor Natuur- en Bosonderzoek (INBO). doi:10.21436/inbor.18840851.
- De Saeger, S., Oosterlynck, P. & Paelinckx, D. (2017). The Biological Valuation Map (BVM): a field-driven survey of land cover and vegetation in the Flemish Region of Belgium. Documents phytosociologiques - Actes du colloque de Saint-Mandé 2012 - Prodrome et cartographie des végétations de France - 2017. Vol. 6: 372-382.

See Also

Other functions involved in processing the 'habitatmap' data source: read_habitatmap_stdized(), read_habitatmap(), read_watersurfaces_hab()

Examples

```
## Not run:
# This example supposes that your working directory or a directory up to 10
# levels above has the 'n2khab_data' folder AND that the latest version of
```

```
# the 'habitatmap_terr'
# data source is present in the default subdirectory.
# In all other cases, this example won't work but at least you can
# consider what to do.

habmap_terr <- read_habitatmap_terr()
habmap_terr$habitatmap_terr_polygons
habmap_terr$habitatmap_terr_types

habmap_terr_noaq <- read_habitatmap_terr(keep_aq_types = FALSE)
habmap_terr_noaq$habitatmap_terr_polygons
habmap_terr_noaq$habitatmap_terr_types

## End(Not run)
```

---

read_habitatquarries *Return the data source* habitatquarries

---

Description

Returns the raw data source habitatquarries as an sf polygon layer in the Belgian Lambert 72 CRS (EPSG-code 31370). Optionally, associated bibliographic references can be returned (arguments references or bibtex).

Usage

```
read_habitatquarries(
 file = file.path(locate_n2khab_data(), "10_raw/habitatquarries/habitatquarries.gpkg"),
  filter_hab = FALSE,
  references = FALSE,
  bibtex = FALSE,
  version = "habitatquarries_2020v1"
)
```

Arguments

file	The absolute or relative file path of the data source. The default follows the data management advice in the vignette on data storage (run vignette("v020_datastorage")). It uses the first n2khab_data folder that is found when sequentially climbing up 0 to 10 levels in the file system hierarchy, starting from the working directory.
filter_hab	If TRUE, only polygons with (known) habitat 8310 are returned.
references	If TRUE, a list is returned with both the sf object (element habitatquarries) and a data frame of bibliographic references (element extra_references).
bibtex	If TRUE, all that happens is bibliographic references being printed to the console, formatted for usage in a BibTeX file (*.bib).
version	Version ID of the data source. Defaults to the latest available version defined by the package.

Details

The data source habitatquarries is a GeoPackage, available at Zenodo, that contains:

- habitatquarries: a spatial polygon layer that corresponds with the presence or absence of the Natura 2000 Annex I habitat type 8310 (Caves not open to the public) in the Flemish Region (and border areas), Belgium;
- extra_references: a non-spatial table of site-specific bibliographic references.

In general, different polygons represent different quarry units with their own internal climatic environment. Units that cross Flemish borders have been split into separate polygons. Exceptionally they may overlap if such units are situated above each other.

Value

Depending on the arguments, one of:

- a simple feature collection of type POLYGON, with attribute variables:
  - polygon_id: a unique number per polygon.
  - unit_id: a unique number for each quarry unit. Quarry units consisting of several polygons (= partly outside the Flemish region) have a number greater than or equal to 100.
  - name: site name.
  - code_orig: original 'habitattype' code in the raw data source habitatquarries.
  - type: habitat type listed in types - in this case either 8310 or missing (NA).
  - extra_reference: site-specific bibliographic reference(s). Values refer to rows in the non-spatial data frame extra_references.
- *if* references = TRUE*:* a list with both the sf object (element habitatquarries) and a data frame of bibliographic references (element extra_references).
- *if* bibtex = TRUE*:* NULL (invisibly).

Examples

```
## Not run:
# This example supposes that your working directory or a directory up to 10
# levels above has the 'n2khab_data' folder AND that the latest version of
# the 'habitatquarries'
# data source is present in the default subdirectory.
# In all other cases, this example won't work but at least you can
# consider what to do.

hq <- read_habitatquarries()
hq
hq2 <- read_habitatquarries(filter_hab = TRUE)
hq2
hq3 <- read_habitatquarries(references = TRUE)
hq3
read_habitatquarries(bibtex = TRUE)

## End(Not run)
```

read_habitatsprings *Return the data source* habitatsprings *as an* sf *point layer*

Description

Returns the raw data source habitatsprings as an sf point layer in the Belgian Lambert 72 CRS (EPSG-code 31370). Optionally, a derived sf object of type-7220-locations can be returned at the population unit level, through aggregation by unit_id.

Usage

```
read_habitatsprings(
  file = file.path(locate_n2khab_data(), "10_raw/habitatsprings/habitatsprings.geojson"),
  filter_hab = FALSE,
  units_7220 = FALSE,
  version = "habitatsprings_2020v2"
)
```

Arguments

file	The absolute or relative file path of the data source. The default follows the data management advice in the vignette on data storage (run vignette("v020_datastorage")). It uses the first n2khab_data folder that is found when sequentially climbing up 0 to 10 levels in the file system hierarchy, starting from the working directory.
filter_hab	If TRUE, only points with (potential) habitat are returned. The default value is FALSE.
units_7220	If TRUE, an sf object of type-7220-locations is returned at the population unit level. To accomplish this, the data source is aggregated by unit_id. Multiple points belonging to the same unit are replaced by their centroid, their area attribute is summed (if all values are known) and for other attributes the maximum value is returned.
version	Version ID of the data source. Defaults to the latest available version defined by the package.

Details

The data source habitatsprings is a GeoJSON file (conforming to the RFC7946 specifications), available at Zenodo. It represents sites that correspond with presence or absence of the Natura 2000 habitat type 7220 (Petrifying springs with tufa formation (*Cratoneurion*)) in springs and streaming water segments in the Flemish Region, Belgium.

Value

A Simple feature collection of type POINT, with attribute variables:

- point_id
- name: site name.

- system_type: environmental typology of 7220: mire, rivulet or unknown (non-7220 types are NA)
- code_orig: original type code in raw habitatsprings.
- type: habitat type listed in types.
- certain: TRUE when the type is certain and FALSE when the type is uncertain.
- unit_id: population unit id for large scale sampling events. Spatially close points have the same value.
- area_m2: area as square meters.
- year: year of field inventory.
- in_sac: logical. Is the site situated within a Special Area of Conservation?
- source: original data source of the record.

Note that the type and system_type variables have implicit NA values (i.e. there is no factor level to represent the missing values). If you want this category to appear in certain results, you can add it as a level with forcats::fct_explicit_na().

With units_7220 = TRUE, variable point_id is dropped and an extra attribute variable nr_of_points is added. It represents the number of points that belong to each unit.

Examples

```
## Not run:
# This example supposes that your working directory or a directory up to 10
# levels above has the 'n2khab_data' folder AND that the latest version of
# the 'habitatsprings'
# data source is present in the default subdirectory.
# In all other cases, this example won't work but at least you can
# consider what to do.

hs <- read_habitatsprings()
hs
hs2 <- read_habitatsprings(units_7220 = TRUE)
hs2

## End(Not run)
```

---

read_habitatstreams *Return the data source* habitatstreams *as an* sf *linestring layer or as a list*

Description

Returns the raw data source habitatstreams (Leyssen et al., 2020) as an sf linestring layer or as a list of two objects: the sf object (CRS: Belgian Lambert 72 (EPSG-code 31370)) plus a data frame with textual explanation of the values of the source_id variable.

Usage

```
read_habitatstreams(
  file = file.path(locate_n2khab_data(), "10_raw/habitatstreams"),
  source_text = FALSE
)
```

Arguments

`file`	The absolute or relative file path of the data source. The default follows the data management advice in the vignette on data storage (run `vignette("v020_datastorage")`). It uses the first `n2khab_data` folder that is found when sequentially climbing up 0 to 10 levels in the file system hierarchy, starting from the working directory.
`source_text`	Logical, defaults to `FALSE`. If `TRUE`, a list is returned (see *Value*).

Value

With `source_text = FALSE` (default): a Simple feature collection of type `LINESTRING`.

With `source_text = TRUE`: a list of two objects:

- `lines`: the same `sf` object as with `source_text = FALSE`.
- `sources`: textual explanation on the values of the `source_id` variable in the `sf` object.

References

Leyssen A., Smeekens V., Denys L. (2020). Indicatieve situering van het Natura 2000 habitattype 3260. Submontane en laaglandrivieren met vegetaties behorend tot het *Ranunculion fluitantis* en het *Callitricho-Batrachion*. Uitgave 2020 (versie 1.7). Rapporten van het Instituut voor Natuur- en Bosonderzoek 2020 (34). Research Institute for Nature and Forest, Brussels. doi:10.21436/inbor.18903609.

Examples

```
## Not run:
# This example supposes that your working directory or a directory up to 10
# levels above has the 'n2khab_data' folder AND that the latest version of
# the 'habitatstreams'
# data source is present in the default subdirectory.
# In all other cases, this example won't work but at least you can
# consider what to do.

library(magrittr)
library(sf)
hs <- read_habitatstreams()
hs
hs2 <- read_habitatstreams(source_text = TRUE)
hs2
all.equal(
  hs %>% st_drop_geometry(),
  hs2$lines %>% st_drop_geometry()
)
```

```
## End(Not run)
```

`read_namelist` *Return the 'namelist' data source as a tibble*

Description

Returns the included data source `namelist` as a `tibble`, by default filtered according to English names and shortnames.

Usage

```
read_namelist(
  path = pkgdatasource_path("textdata/namelist", ".yml"),
  file = "namelist",
  lang = "en"
)
```

Arguments

`path`	Location of the data source. The default is to use the location of the data source as delivered by the installed package.
`file`	The filename of the data source, without extension. The default is to use the file delivered by the installed package.
`lang`	An IETF BCP 47 language tag, such as `"en"` or `"nl"`, to specify the language of `name` and `shortname` to be returned in the tibble. If `lang = "all"`, the full `namelist` tibble is returned, i.e. containing all languages.

Details

`namelist` is a data source in the vc-format which provides names and (optionally) shortnames for IDs/codes used in other data sources.

`read_namelist()` reads it and returns it as a `tibble`. A tibble is a data frame that makes working in the tidyverse a little easier. By default, the data version delivered with the package is used and only English names (`lang = "en"`) are returned.

Value

The `namelist` data frame as a `tibble`, filtered according to the `lang` argument. See `namelist` for documentation of the tibble's contents.

Recommended usage

```
read_namelist()
```

```
read_namelist(lang = "nl")
```

See Also

namelist

Other reading functions for n2khab-referencelists: read_env_pressures(), read_types()

Examples

```
read_namelist()
read_namelist(lang = "nl")
```

---

read_raster_runif *Return the data source* raster_runif *as a SpatRaster*

Description

The raster_runif data source covers Flanders and the Brussels Capital Region and has a resolution of 32 meters. The raster cells with non-missing values match the value-cells of the GRTSmaster_habitats data source (see read_GRTSmh) with a small buffer added. Every raster cell has a random value between 0 and 1 according to the uniform distribution. The coordinate reference system is 'BD72 / Belgian Lambert 72' (EPSG-code 31370).

Usage

```
read_raster_runif(
  file = file.path(locate_n2khab_data(), "10_raw/raster_runif/raster_runif.tif"),
  version = "raster_runif_v1"
)
```

Arguments

file	The absolute or relative file path of the data source. The default follows the data management advice in the vignette on data storage (run vignette("v020_datastorage")). It uses the first n2khab_data folder that is found when sequentially climbing up 0 to 10 levels in the file system hierarchy, starting from the working directory.
version	Version ID of the data source. Defaults to the latest available version defined by the package.

Details

The raster_runif data source is a GeoTIFF file (available at Zenodo).

The R-code for creating the raster_runif data source can be found in the n2khab-preprocessing repository.

Value

A SpatRaster.

If the package is configured to use the raster package (see n2khab_options()), a RasterLayer is returned instead.

Examples

```
## Not run:
# This example supposes that your working directory or a directory up to 10
# levels above has
# the 'n2khab_data' folder AND that the latest version of the
# 'raster_runif' data source is present in the default subdirectory.
# In all other cases, this example won't work but at least you can consider
# what to do.
r <- read_raster_runif()
r

## End(Not run)
```

`read_shallowgroundwater` *Return the data source* `shallowgroundwater` *as an* `sf` *multipolygon layer*

Description

Returns the raw data source `shallowgroundwater` as an `sf` multipolygon layer. The coordinate reference system is 'BD72 / Belgian Lambert 72' (EPSG-code 31370).

Usage

```
read_shallowgroundwater(
  file = file.path(locate_n2khab_data(),
    "10_raw/shallowgroundwater/shallowgroundwater.gpkg")
)
```

Arguments

`file`	The absolute or relative file path of the data source. The default follows the data management advice in the vignette on data storage (run `vignette("v020_datastorage")`). It uses the first `n2khab_data` folder that is found when sequentially climbing up 0 to 10 levels in the file system hierarchy, starting from the working directory.

Details

The data source `shallowgroundwater` represents the areas in the Flemish region of Belgium where the mean lowest groundwater level (MLW; Knotters & Van Walsum, 1997; Van Heesen, 1970) is estimated to be less than approximately 2 m below soil surface (hence, 'shallow' groundwater). Groundwater dependent species and communities can be expected to be present mostly within these areas.

The data source is a GeoPackage, available at Zenodo, and contains a single spatial multipolygon layer `shallowgroundwater`.

The data source has been compiled from subsets of various other spatial data sources. This process is described in more detail at Zenodo. The R code for finishing the `shallowgroundwater` data source, starting from an intermediate result, can be found in the n2khab-preprocessing repository.

Value

A Simple feature collection of geometry type `MULTIPOLYGON`.

All attribute variables are logical variables referring to a data source (subset) or procedure that has contributed to the `shallowgroundwater` data source. If a variable is `TRUE` for a multipolygon, then the related data source (subset) or procedure has contributed to the multipolygon. For one multipolygon, several variables can be `TRUE` at the same time. Each combination of values occurs in only one multipolygon.

The attribute variables listed below are described in more detail at Zenodo; their description typically refers to the meaning of `TRUE`:

- `geomorph_wcoast`: mainly concerns dune slacks, mud flats and salt marshes. *Derived from: Cosyns et al. (2019)*
- `anthrop_gwdep`: zones located within a 100 m buffer around (almost) everywhere groundwater dependent habitat types (or regionally important biotopes) ánd situated within zones classified as 'anthropogenic' areas within the soil map. *Derived from:* `soilmap_simple` *and* `habitatmap_terr` *data sources*
- `narrowanthrop_gwdep`: narrow zones classified as 'anthropogenic' areas within the soil map that include (almost) everywhere groundwater dependent habitat types (or regionally important biotopes). *Derived from:* `soilmap_simple` *and* `habitatmap_terr` *data sources*
- `drainage`: soils that are at least moderately gleyic or wet. *Derived from:* `soilmap_simple` *data source*
- `dunes_gwdep`: zones located within a 100 m buffer around (almost) everywhere groundwater dependent habitat types (or regionally important biotopes) ánd situated within zones classified as 'dunes' areas within the soil map. *Derived from:* `soilmap_simple` *and* `habitatmap_terr` *data sources*
- `peat_profile`: variant of the soil profile indicates a superficial peaty cover, mostly on gleyic or permanently water saturated soil with or without profile development (`(v)`), eventually combined with strong anthropogenic influence (`(o)`). *Derived from:* `soilmap_simple` *data source*
- `peat_substr`: soil substrate (layer underlying superficial layer, and lithologically diverging from it) consists of peat material starting at small (less than 75 cm; `v`) or moderate depths (75-125 cm; `(v)`), or a combination of the previous (`v-`). *Derived from:* `soilmap_simple` *data source*
- `peat_parentmat`: parent material contains a mixture of at least 30% of peaty material. *Derived from:* `soilmap_simple` *data source*
- `peat_texture`: soil consists of plain peat material. *Derived from:* `soilmap_simple` *data source*
- `phys_system`: polygons designated as seepage areas where groundwater is supposed to gather after having infiltrated elsewhere (infiltration areas) and being transported through the landscape (passage areas). *Derived from: Lhermitte & Honnay (1994)*

- `zwin`: the contour of the Zwin Nature Reserve in the most eastern part of the Flemish coastal area. *Derived from: Open Street Map, consulted 2021-10-06 by QGIS plugin QuickOSM*
- `habitat_1130`: polygons located within estuaries (habitat type `1130`). *Derived from:* `habitatmap` *data source*
- `gwdepth_coast`: locations with estimated average lowest groundwater table less than 2.5 m below soil surface (shallow groundwater), based on interpolation of measured groundwater levels in areas along the Flemish coast with sufficient gauge densities. *Derived from: the Watina database of the Research Institute for Nature and Forest (INBO)*
- `gwdepth_local`: mean lowest groundwater level less than 2 m below soil surface (MLW) in a large military training site for which no information is available in the soil map of Flanders. *Derived from: Batelaan et al. (2012)*
- `seepage`: area with modelled seepage fluxes of at least 0.8 mm/day in the central part of eastern Flanders. *Derived from: Batelaan & De Smedt (1994)*
- `peat_survey`: local peaty zones evaluated by simple measurements of the depth of plain and superficial peaty soil layers at regular intervals (about 20 m). *Derived from: local inventories executed or compiled by the Research Institute for Nature and Forest (INBO)*
- `duneslack`: polygons with dune slack vegetations along the Flemish coast that typically imply shallow groundwater levels. *Derived from: Provoost et al. (2020)*

References

- Batelaan O., De Becker P., El-Rawy M., Herr C., Schneidewind, U. (2012). Doorrekenen van maatregelen voor herstel van vochtige heidevegetaties op het Schietveld van Houthalen-Helchteren via grondwatermodellering. Vrije Universiteit Brussel/Instituut voor Natuur en Bosonderzoek.
- Batelaan O., De Smedt F. (1994). Regionale grondwaterstroming rond een aantal kwelafhankelijke natuurgebieden. Vrije Universiteit Brussel.
- Cosyns E., Bollengier B., Provoost S. (2019). Masterplan en juridische basis voor grensoverschrijdende samenwerking en bescherming als een transnationaal natuurpark van de duinen tussen Dunkerque (Frankrijk) en Westende (België). Partim Masterplan. Rapport in opdracht van Agentschap Natuur en Bos, Conservatoire de l'espace littoral et des rivages lacustres, Conseil Général Département du Nord. https://www.natuurenbos.be/sites/default/files/inserted-files/masterplan_flandre_ned20200210def.pdf
- Knotters M. & van Walsum P.E.V. (1997). Estimating fluctuation quantities from time series of water-table depths using models with a stochastic component. Journal of Hydrology 197 (1): 25–46. doi:10.1016/S00221694(96)032787.
- Lhermitte K., Honnay O. (1994). Kartering van het Fysisch Systeem en de Ruimtelijke structuren in Vlaanderen op schaal 1 : 50 000. Stichting Plattelandsbeleid 1994, Boerenbond, Leuven.
- Provoost S., Van Gompel W. & Vercruysse E. (2020). Beheerevaluatie kust. Eindrapport 2015-2019. Rapporten van het Instituut voor Natuur- en Bosonderzoek 2020 (18). Instituut voor Natuur- en Bosonderzoek, Brussel. doi:10.21436/inbor.18039583.
- Van Heesen H.C. (1970). Presentation of the seasonal fluctuation of the water table on soil maps. Geoderma 4 (3): 257–278. doi:10.1016/00167061(70)900066.

See Also

Other functions returning environmental data sets: `read_soilmap()`, `read_watercourse_100mseg()`, `read_watersurfaces()`

Examples

```
## Not run:
shallowgroundwater <- read_shallowgroundwater()
shallowgroundwater

## End(Not run)
```

`read_soilmap` *Return the* `soilmap` *or* `soilmap_simple` *data source as an* `sf` *multi-polygon layer*

Description

Returns either the raw data source `soilmap` or (by default) the processed data source `soilmap_simple` as a standardized `sf` multipolygon layer (tidyverse-styled, internationalized) in the Belgian Lambert 72 CRS (EPSG-code 31370). Given the size of these data sources (especially the raw one), this function takes a bit longer than usual to run.

Usage

```
read_soilmap(
  file = file.path(locate_n2khab_data(),
    "20_processed/soilmap_simple/soilmap_simple.gpkg"),
  file_raw = file.path(locate_n2khab_data(), "10_raw/soilmap"),
  use_processed = TRUE,
  version_processed = "soilmap_simple_v2",
  standardize_coastalplain = FALSE,
  simplify = FALSE,
  explan = FALSE
)
```

Arguments

`file`	The absolute or relative file path of the *processed* data source `soilmap_simple`. Used only if `use_processed = TRUE` (= default). The default value follows the data management advice in the vignette on data storage (run `vignette("v020_datastorage")`). It uses the first `n2khab_data` folder that is found when sequentially climbing up 0 to 10 levels in the file system hierarchy, starting from the working directory.
`file_raw`	Same as `file`, to define the filepath of the *raw* datasource `soilmap`. Used only if `use_processed = FALSE`.

use_processed
: Logical. If TRUE (the default), load and return the processed data source soilmap_simple, instead of the raw data source soilmap. The central layer of soilmap_simple can be manually generated by reading the raw soilmap data source with standardize_coastalplain=TR simplify=TRUE and explan=FALSE, but this takes some time.

version_processed
: Version ID of the soilmap_simple data source. Only used with use_processed = TRUE (the default). Defaults to the latest available version defined by the package.

standardize_coastalplain
: Logical. Only applied with use_processed = FALSE. If TRUE, fill the values of the morphogenetic substrate, texture and drainage variables (bsm_mo_substr, bsm_mo_tex and bsm_mo_drain + their _explan counterparts) where possible, *for features with a geomorphological soil type code*. This largely applies to features in the 'coastal plain' area.

  - To derive morphogenetic texture and drainage levels from the geomorphological soil types, a conversion table by Bruno De Vos & Carole Ampe is applied (for earlier work on this, see Ampe 2013).
  - Substrate classes are copied over from bsm_ge_substr into bsm_mo_substr (bsm_ge_substr already follows the categories of bsm_mo_substr).
  - A logical variable is added to the output to mark conversions (see section *Value*).

  These steps coincide with the approach that was taken to construct bsm_mo_soilunitype in the raw data source.

simplify
: Logical. Only applied with use_processed = FALSE. If TRUE, only a limited number of variables that are most useful for analytical work are returned.

explan
: Logical, defaults to FALSE. Should the _explan variables accompanying bsm_mo_xxx variables be returned in the *simplified* result? If use_processed = FALSE: only has effect if simplify=TRUE. (With simplify=FALSE, the _explan variables are always returned.) If use_processed = TRUE: is applied in returning soilmap_simple.

Details

The raw data source is published at DOV (Databank Ondergrond Vlaanderen) and is discussed by Van Ranst & Sys (2000) and Dudal et al. (2005). A 'pure' (single) dataformat of the raw data source (no metadatafiles etc.) has also been stored (with versioning) at Zenodo (doi:10.5281/zenodo.3387007) - which we refer to as the soilmap data source - in order to support the read_soilmap() function and to sustain long-term workflow reproducibility.

The processed data source soilmap_simple is a GeoPackage, available at Zenodo.

Note that factors are generated with implicit NA values (i.e. there is no factor level to represent the missing values). If you want this category to appear in certain results, you can convert such variables with forcats::fct_explicit_na().

In case the raw data source soilmap is used (use_processed = FALSE), it is possible to manually perform the standardization for coastal plain features and/or the simplification, both of which were applied in the soilmap_simple data source. See *Arguments* for more information.

See R-code in the n2khab-preprocessing repository for the creation of the soilmap_simple data source from the soilmap data source.

Value

A Simple feature collection of geometry type MULTIPOLYGON, representing either the processed data source soilmap_simple (default) or the raw data source soilmap.

Besides the standardization for the coastal plain areas, soilmap_simple contains only a subset of the soilmap variables (marked with an asterisk below).

The soilmap attribute variables all start with prefix bsm_ (referring to the 'Belgian soil map'), in order to distinguish from similar attributes derived from other maps or field observations.

Most attributes represent categories and are returned as factors. When a variable is a one-to-one translation of another (e.g. code vs. explanation), the order of factor levels is aligned.

Three types of data frame variables are returned when reading soilmap:

- **variables with** mo_ **in their name**: their categories follow the Belgian Morphogenetic System.
  - With standardize_coastalplain = FALSE, these are only available *outside the coastal plain areas* except for bsm_mo_soilunitype (which is standardized already in the raw data source).
- **variables with** ge_ **in their name**: their categories follow the Belgian Geomorphological System. (Note however, that bsm_ge_substr does follow the Belgian Morphogenetic System as well.)
  - Values are typically available *within the coastal plain areas*, but some geomorphological soil types (starting with letter 0) have a wider distribution across Flanders.
  - They are *not* included in soilmap_simple.
  - A special variable is bsm_ge_typology, which is TRUE if bsm_soiltype follows the geomorphological typology, and FALSE otherwise.
- **variables without** mo_ **or** ge_ **in their name** are:
  - either *system-agnostic* metadata (first two + last four variables: bsm_poly_id, bsm_map_id, bsm_map_url, bsm_book_url, bsm_detailmap_url, bsm_profloc_url),
  - or *mixed* (representing mo_ categories within and ge_ categories outside coastal plains): the other ones, like bsm_region, bsm_legend, bsm_soiltype and bsm_soilseries.
  - A special variable is bsm_converted, returned only if standardize_coastalplain = TRUE.

Many variables have a 'counterpart variable' with suffix _explan: they provide a more elaborate textual explanation. They are not listed below.

Short explanation of attributes is given below. More elaborate explanations can be found in the references and in metadata at DOV.

1. Meaning of the main non-metadata variables:
   - bsm_region (*): name of the region
   - bsm_ge_region: code of the region within the coastal plain area
   - bsm_legend: generalised (simplified) legend key (37 levels)
   - bsm_legend_title and bsm_legend_explan: the legend keys and text of Van Ranst & Sys (2000) (833 and 622 levels, respectively)
   - bsm_soiltype: the soil type of the Belgian soil map (mixed nature: morphogenetic & geomorphological codes). bsm_soiltype_id represents a numeric code for each level.

- bsm_ge_typology: Logical. Does the soiltype code follow the geomorphological typology?
- bsm_soiltype_region: bsm_soiltype, followed by a code representing bsm_region
- bsm_soilseries: either the morphogenetic soil series (outside the coastal plain areas), which is the three core characters of bsm_soiltype, or just bsm_soiltype if the latter has a geomorphological code.
- bsm_converted (*): Logical. Were morphogenetic texture and drainage variables (bsm_mo_tex and bsm_mo_drain) derived from a conversion table? This is equivalent with the question: does bsm_mo_soilunitype differ from bsm_soiltype? Value TRUE is largely confined to the 'coastal plain' areas. Only returned if standardize_coastalplain = TRUE. (Note: the variable is not included in version soilmap_simple_v1.)
- bsm_mo_soilunitype (*): as bsm_soiltype, but applying morphogenetic codes within the coastal plain areas in most cases (see the standardize_coastalplain argument for more information about this conversion)
- bsm_mo_substr (*), bsm_ge_substr: code of the soil substrate
- bsm_mo_tex (*): code of the soil texture category
- bsm_mo_drain (*): code of the soil drainage category
- bsm_mo_prof (*): code of the soil profile category
- bsm_mo_parentmat (*): code of a variant regarding the parent material
- bsm_mo_profvar (*): code of a variant regarding the soil profile
- bsm_mo_phase: code of the soil phase (i.e. additional soil properties). They are explained in the book that accompanies the specific analog map identified by bsm_map_id.
- bsm_ge_series: the geomorphological soil series
- bsm_ge_subseries: the geomorphological soil subseries

2. Meaning of the metadata variables:
   - bsm_poly_id (*): unique polygon ID (numeric)
   - bsm_map_id: code of the analog map covering this area
   - bsm_map_url: hyperlink to the scanned analog map scale 1:20000 (pdf), identified by bsm_map_id
   - bsm_bookurl: hyperlink to the scanned book (pdf), accompanying the analog map identified by bsm_map_id
   - bsm_detailmap_url: hyperlink to the scanned maps at scale 1:5000 (zip-file with jpg files) belonging to the map identified by bsm_map_id
   - bsm_profloc_url: hyperlink to the scanned maps with the profile locations (zip-file with jpg files) belonging to the map identified by bsm_map_id

(*) Included in the soilmap_simple data source.

References

- Ampe C. (2013). Databank aardewerk Vlaanderen 2010. Omzetten (zeer) oude legende bodemkartering naar legende bodemkaart Kuststreek. Vlaamse Landmaatschappij Regio West, Bruges, 45 p.
- Dudal R., Deckers J., Van Orshoven J. & Van Ranst E. (2005). Soil survey in Belgium and its applications. In: Bullock P., Jones R.J.A., Montanarella L. (editors). Soil Resources of Europe. Office for Official Publications of the European Communities, Luxembourg, p. 63–71. URL: http://hdl.handle.net/1854/LU-368514.

- Van Ranst E. & Sys C. (2000). Eenduidige legende van de digitale bodemkaart van Vlaanderen (schaal 1: 20000). Universiteit Gent, Laboratorium voor Bodemkunde, Ghent, 361 p. URL: http://hdl.handle.net/1854/LU-125899.

See Also

Other functions returning environmental data sets: read_shallowgroundwater(), read_watercourse_100mseg(), read_watersurfaces()

Examples

```
## Not run:
# This example supposes that your working directory or a directory up to 10
# levels above has the 'n2khab_data' folder AND that the latest version of
# the 'soilmap_simple'
# data source is present in the default subdirectory.
# In all other cases, this example won't work but at least you can
# consider what to do.

library(dplyr)
soilmap_simple <- read_soilmap()
soilmap_simple
soilmap_simple %>%
  filter(!is.na(bsm_mo_substr)) %>%
  glimpse()
soilmap_simple %>%
  filter(bsm_converted) %>%
  glimpse()

## End(Not run)
```

---

read_types *Return the 'types' data source as a tibble with names & shortnames*

Description

Returns the included data source types as a tibble. Names and shortnames from namelist are added, in English by default.

Usage

```
read_types(
  path = pkgdatasource_path("textdata/types", ".yml"),
  file = "types",
  file_namelist = "namelist",
  lang = "en"
)
```

Arguments

path	Location of the data sources types and namelist. The default is to use the location of the data sources as delivered by the installed package.
file	The filename of the types data source, without extension. The default is to use the file delivered by the installed package.
file_namelist	The filename of the namelist data source, without extension. The default is to use the file delivered by the installed package.
lang	An IETF BCP 47 language tag, such as "en" or "nl", to specify the language of names & shortnames to be returned in the tibble.

Details

types is a data source in the vc-format which provides a checklist of types, represented by their **current** codes, together with several attributes. A 'type' refers to either a (main) habitat type, a habitat subtype or a regionally important biotope (RIB).

read_types() reads the types data source, adds names + shortnames and returns it as a tibble. A tibble is a data frame that makes working in the tidyverse a little easier. By default, the data version delivered with the package is used and English names (lang = "en") are returned for types, attributes and tags.

Note that factors are generated with implicit NA values (i.e. there is no factor level to represent the missing values). If you want this category to appear in certain results, you can convert such variables with forcats::fct_explicit_na().

Value

The types data frame as a tibble, with names & shortnames added for types, attributes and tags according to the lang argument. See types for documentation of the data-source's contents. See namelist for the link between codes or other identifiers and the corresponding names (and shortnames).

The added names and shortnames are represented by the following variables:

- type_name
- type_shortname
- typeclass_name
- hydr_class_name
- hydr_class_shortname
- groundw_dep_name
- groundw_dep_shortname
- flood_dep_name
- flood_dep_shortname
- tag_1_name
- tag_1_shortname
- tag_2_name

- tag_2_shortname
- tag_3_name
- tag_3_shortname

Except for the tags, the names and shortnames are factors with their level order according to that of the corresponding attribute.

Recommended usage

```
read_types()
read_types(lang = "nl")
```

See Also

types

Other reading functions for n2khab-referencelists: read_env_pressures(), read_namelist()

Examples

```
read_types()
read_types(lang = "nl")
```

---

read_watercourse_100mseg

Return the processed data source watercourse_100mseg

Description

Returns the data source watercourse_100mseg as a list of two sf objects:

- lines (LINESTRING geometry): represents line segments of length 100 m derived from the raw watercourses data source;
- points (POINT geometry): represents the *downstream* endpoints of all segments ('downstream' as defined in watercourses).

Optionally, only one of these objects is returned. The coordinate reference system is 'BD72 / Belgian Lambert 72' (EPSG-code 31370).

Usage

```
read_watercourse_100mseg(
  file = file.path(locate_n2khab_data(),
    "20_processed/watercourse_100mseg/watercourse_100mseg.gpkg"),
  element = NULL,
  version = "watercourse_100mseg_20200807v1"
)
```

Arguments

`file`	The absolute or relative file path of the data source. The default follows the data management advice in the vignette on data storage (run `vignette("v020_datastorage")`). It uses the first `n2khab_data` folder that is found when sequentially climbing up 0 to 10 levels in the file system hierarchy, starting from the working directory.
`element`	Optional string. The string must be one of two possible values: `"lines"` or `"points"`. If set, either the `lines` or the `points` object will be returned, respectively.
`version`	Version ID of the data source. Defaults to the latest available version defined by the package.

Details

The data source `watercourse_100mseg` represents all officially known watercourses of the Flemish Region as line segments of 100 m (or < 100 m, for the most upstream segment of a watercourse). The data source can be used as a base layer of statistical **population units** (line segments) and corresponding anchor points, in the design of monitoring and research of watercourses.

The data source is a GeoPackage, available at Zenodo, and contains two spatial layers:

- `watercourse_100mseg_lines`: the line segments;
- `watercourse_100mseg_points`: the corresponding downstream endpoints.

Both layers have the same number of rows and share the same attributes.

The data source was derived from the raw `watercourses` data source as follows:

1. each line ('watercourse') of `watercourses` is split into segments of 100 m, where the remaining segment of < 100 m (per original line) is situated most upstream. For this step, the direction of the lines has been reverted (in `watercourses` the direction is from upstream to downstream). A unique rank number is assigned to each segment, as well as the VHAG code from the corresponding line in `watercourses`.
2. the downstream endpoint of each segment is located, and assigned the same attributes (`rank` and `vhag_code`).

The R and GRASS code for creating the `watercourse_100mseg` data source can be found in the n2khab-preprocessing repository.

Value

By default, a list of two `sf` objects (see 'Description'). The `lines` and the `points` objects have the same number of rows. They share the same attributes:

`rank` A unique, incremental number for each segment/endpoint. It just reflects the downstream-to-upstream order of segments within each original line.

`vhag_code` The VHAG code from the raw `watercourses` data source. It distinguishes the different watercourses, so it is common to all segments/points that belong to the same watercourse.

Optionally, only one of these `sf` objects is returned.

See Also

Other functions returning environmental data sets: `read_shallowgroundwater()`, `read_soilmap()`, `read_watersurfaces()`

Examples

```
## Not run:
# This example supposes that your working directory or a directory up to 10
# levels above has the 'n2khab_data' folder AND that the latest version of the
# 'watercourse_100mseg' data source is present in the default subdirectory.
# In all other cases, this example won't work but at least you can
# consider what to do.

(lines <- read_watercourse_100mseg(element = "lines"))
(points <- read_watercourse_100mseg(element = "points"))
str(read_watercourse_100mseg(), give.attr = FALSE)

## End(Not run)
```

---

read_watersurfaces *Return the data source* watersurfaces *as an* sf *polygon layer*

Description

Returns the raw data source `watersurfaces` (Scheers et al., 2022) as a standardized `sf` polygon layer (tidyverse-styled, internationalized) in the Belgian Lambert 72 CRS (EPSG-code 31370).

Usage

```
read_watersurfaces(
  file = NULL,
  extended = FALSE,
  fix_geom = FALSE,
  version = c("watersurfaces_v1.2", "watersurfaces_v1.1", "watersurfaces_v1.0")
)
```

Arguments

`file`	Optional string. An absolute or relative file path of the data source. If left `NULL`, the default follows the data management advice in the vignette on data storage (run `vignette("v020_datastorage")`). It uses the first `n2khab_data` folder that is found when sequentially climbing up 0 to 10 levels in the file system hierarchy, starting from the working directory.
`extended`	Logical. Should names or explanations of codes be added as extra variables in the result? Currently only applies to `wfd_type` and `connectivity`; if `TRUE`, the variables `wfd_type_name` and `connectivity_name` are added. Defaults to `FALSE`.

fix_geom	Logical. Should invalid or corrupt geometries be fixed in the resulting sf object in order to make them valid? This prevents potential problems in geospatial operations, but beware that fixed geometries are different from the original ones. sf::st_make_valid() is used to fix geometries (with GEOS as backend). Defaults to FALSE.
version	Version ID of the data source. Defaults to the latest available version defined by the package.

Details

If file is not specified, the function will try to read the file in the default folder for data storage (as described in the data management advice in the vignette (run vignette("v020_datastorage"))). If you want to use another file or file structure than the default data storage, you can specify your own file. In both cases: always make sure to specify the correct version, that is the version corresponding to the file (note that the version defaults to the latest version).

See Scheers et al. (2022) for an elaborate explanation of the data source and its attributes.

Value

A Simple feature collection of type POLYGON, sorted by polygon_id, with the following variables (not mentioning extra 'name' variables for extended = TRUE):

- polygon_id: code of the polygon;
- wfd_code: optional; Flemish code of the water body with regard to the Water Framework Directive (WFD);
- hyla_code: optional; code of the watersurface according to the Flemish working group 'Hyla', a working group on amphibians & reptiles;
- name: name of the watersurface;
- area_name: name of the area;
- wfd_type: type code according to the Flemish WFD typology (Denys, 2009);
- wfd_type_certain: Logical. Is there high confidence about the wfd_type determination?
- depth_class: class of water depth;
- connectivity: connectivity class;
- usage: usage class;
- water_level_management: water-level management class (not in older versions).

References

- Denys L. (2009). Een a posteriori typologie van stilstaande, zoete wateren in Vlaanderen. Rapporten van het Instituut voor Natuur- en Bosonderzoek INBO.R.2009.34. Instituut voor Natuur- en Bosonderzoek, Brussel.
- Scheers K., Smeekens V., Wils C., Packet J., Leyssen A. & Denys L. (2022). Watervlakken versie 1.2: Polygonenkaart van stilstaand water in Vlaanderen. Uitgave 2022. Instituut voor Natuur- en Bosonderzoek. doi:10.21436/inbor.87014272.

See Also

Other functions involved in processing the 'watersurfaces' data source: `read_watersurfaces_hab()`

Other functions returning environmental data sets: `read_shallowgroundwater()`, `read_soilmap()`, `read_watercourse_100mseg()`

Examples

```
## Not run:
# This example supposes that your working directory or a directory up to 10
# levels above has the 'n2khab_data' folder AND that the latest version of
# the 'watersurfaces' data source is present in the default subdirectory.
# In all other cases, this example won't work but at least you can
# consider what to do.

ws <- read_watersurfaces()
ws
summary(ws)

ws_valid <- read_watersurfaces(fix_geom = TRUE)
ws_valid

all(sf::st_is_valid(ws))
all(sf::st_is_valid(ws_valid))

ws2 <- read_watersurfaces(extended = TRUE)
ws2

## End(Not run)
```

---

read_watersurfaces_hab

Return the data source `watersurfaces_hab` *as a list of two objects*

Description

`read_watersurfaces_hab` returns the data source `watersurfaces_hab` as a list of two objects:

- `watersurfaces_polygons`: an sf object in the Belgian Lambert 72 CRS (EPSG-code 31370) with all polygons that contain standing water types (habitat or RIB).
- `watersurfaces_types`: a tibble with information on the standing water types (HAB1, HAB2,..., HAB5) that occur within each polygon of `watersurfaces_polygons`.

Usage

```
read_watersurfaces_hab(
  file = file.path(locate_n2khab_data(),
    "20_processed/watersurfaces_hab/watersurfaces_hab.gpkg"),
```

```
  interpreted = FALSE,
 version = c("watersurfaces_hab_v4", "watersurfaces_hab_v3", "watersurfaces_hab_v2",
    "watersurfaces_hab_v1")
)
```

Arguments

file	The absolute or relative file path of the data source. The default follows the data management advice in the vignette on data storage (run `vignette("v020_datastorage")`). It uses the first `n2khab_data` folder that is found when sequentially climbing up 0 to 10 levels in the file system hierarchy, starting from the working directory.
interpreted	If `TRUE`, the interpreted subtype is provided when the subtype is missing. This only applies to type 3130. When the subtype is missing for 3130, we interpret it as 3130_aom.
version	Version ID of the data source. Defaults to the latest available version defined by the package.

Details

The data source `watersurfaces_hab` is a combination of `habitatmap_stdized` (see `read_habitatmap_stdized`) and the watersurface map of Flanders. It contains all standing water types in Flanders.

The data source `watersurfaces_hab` is a GeoPackage, available at Zenodo, that contains:

- `watersurfaces_hab_polygons`: a spatial layer with all polygons that contain standing water types listed in `types`.
- `watersurfaces_hab_types`: a table in which every row corresponds with a combination of polygon and type.

The polygons with 2190_a habitat (dune slack ponds) are generated by selecting all watersurface polygons that overlap with dune habitat polygons (21xx) of the standardized habitat map.

For each of the other considered habitat types (31xx and rbbah) we select the watersurface polygons that overlap with the selected habitat type polygons of the standardized habitat map. We also select polygons of the standardized habitat map that contain standing water types but do not overlap with any watersurface polygon of the watersurface map.

The R-code for creating the `watersurfaces_hab` data source can be found in the n2khab-preprocessing repository.

Value

A list of two objects:

- `watersurfaces_polygons`: an sf object of standing water polygons with four attribute variables:
  - `polygon_id`
  - `polygon_id_ws`: id for the polygon in the `watersurface map`
  - `polygon_id_habitatmap`: id's of all overlapping polygons of `habitatmap_stdized` that contain standing water habitat. The different id's are separated by '+'.

  - description_orig: descriptions of all overlapping polygons of habitatmap_stdized that contain standing water habitat. The different descriptions are separated by '+'.

- watersurfaces_types: a tibble with following variables:
  - polygon_id
  - type: habitat or RIB type listed in types.
  - certain: TRUE when the type is certain and FALSE when the type is uncertain.
  - code_orig: original type code in raw habitatmap.

References

- Leyssen A., Scheers K., Smeekens V., Wils C., Packet J., De Knijf G. & Denys L. (2020). Watervlakken versie 1.1: polygonenkaart van stilstaand water in Vlaanderen. Uitgave 2020. Rapporten van het Instituut voor Natuur- en Bosonderzoek 2020 (40). Instituut voor Natuur en Bosonderzoek, Brussel. doi:10.21436/inbor.19088385.
- De Saeger, S., Guelinckx, R., Oosterlynck, P., De Bruyn, A., Debusschere, K., Dhaluin, P., Erens, R., Hendrickx, P., Hennebel, D., Jacobs, I., Kumpen, M., Op De Beeck, J., Spanhove, T., Tamsyn, W., Van Oost, F., Van Dam, G., Van Hove, M., Wils, C., Paelinckx, D. (2020). Biologische Waarderingskaart en Natura 2000 Habitatkaart, uitgave 2020. (Rapporten van het Instituut voor Natuur- en Bosonderzoek; Nr. 35). Instituut voor Natuur- en Bosonderzoek (INBO). doi:10.21436/inbor.18840851.

See Also

Other functions involved in processing the 'habitatmap' data source: read_habitatmap_stdized(), read_habitatmap_terr(), read_habitatmap()

Other functions involved in processing the 'watersurfaces' data source: read_watersurfaces()

Examples

```
## Not run:
# This example supposes that your working directory or a directory up to 10
# levels above has the 'n2khab_data' folder AND that the latest version of
# the 'watersurfaces_hab'
# data source is present in the default subdirectory.
# In all other cases, this example won't work but at least you can
# consider what to do.

wsh <- read_watersurfaces_hab()
wsh_polygons <- wsh$watersurfaces_polygons
wsh_types <- wsh$watersurfaces_types
wsh_polygons
wsh_types

## End(Not run)
```

types *Documentation of included data source 'types'*

Description

'types' is a data source in the vc-format which provides a checklist of types, represented by their **current** codes, together with several attributes. A 'type' refers to either a (main) habitat type, a habitat subtype or a regionally important biotope (RIB). The codes of types, typeclasses and further attributes and tags are explained in the data source `namelist` (which can accommodate multiple languages).

Format

A vc-formatted data source. As such, it corresponds to a data frame with several variables:

type Code of the type, as a factor. This is the ID for use in diverse workflows and datasets. Names in multiple languages are to be found in `namelist`. Only *currently active* codes are kept in this list, in order to avoid confusion (this especially relates to habitat subtypes and RIBs). Contains no duplicates!

typelevel A factor that labels the type as either `"main_type"` or `"subtype"`.

main_type The main type that corresponds with `type`, as a factor. Each type is either a subtype of a main type, or is a main type itself. This is indicated by `typelevel`.

typeclass A code explained by `namelist`, corresponding to the typeclass. Is a factor.

hydr_class A code explained by `namelist`, corresponding to the hydrological class. Is a factor.

groundw_dep A code explained by `namelist`, corresponding to the groundwater dependency category. Is a factor.

flood_dep A code explained by `namelist`, corresponding to the flood dependency category. Is a factor. Note that flood dependency is only defined for (semi-)terrestrial types, hence for aquatic types (hydrological class `HC3`) it is `NA`.

tag_1 Optional tag, e.g. a categorization ID explained by `namelist`. Currently used to indicate subcategories within a few typeclasses, or to differentiate between lotic and lentic aquatic types.

tag_2 Optional tag, e.g. a categorization ID explained by `namelist`.

tag_3 Optional tag, e.g. a categorization ID explained by `namelist`.

Typical way of loading

```
read_types()
read_types(lang = "nl")
```

Corresponding datafiles in the installed package

```
textdata/types.csv
textdata/types.yml
```

Source

Most information comes from this googlesheet. Currently, the googlesheet and the data source are both kept up-to-date. However only the 'types' data source is under version control.

The source for the hydrological class attribute is a vc-formatted file stored in the package source code. It is read by the 'generate_textdata' bookdown project which generates the 'types' data source. The referred vc-formatted file was derived from a yet unpublished database on the interrelations between types, hydrological classes, environmental compartments and their characteristics, and environmental pressures.

The source for the groundwater and flood dependency attributes is this googlesheet. Currently, the googlesheet and the data source are both kept up-to-date. However only the 'types' data source is under version control.

See Also

`read_types`

Other n2khab-referencelists: `env_pressures`, `namelist`

Index