Package 'neonstore'

October 27, 2020

Title NEON Data Store

Version 0.3.2

Description The National Ecological Observatory Network (NEON) provides access to its numerous data products through its REST API, <https://data.neonscience.org/data-api/>. This package provides a high-level user interface for downloading and storing NEON data products. While each of NEON data products consist of hundreds or thousands of individual files. Unlike 'neonUtilities', this package will avoid repeated downloading, provides persistent storage, and improves performance. This package does not provide expose interactions with the individual low-level NEON API endpoints.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

Imports DBI, duckdb, vroom, rappdirs, httr, R.utils, zip, jsonlite, curl, openssl, tibble, digest, progress, tools, utils

RoxygenNote 7.1.1

Suggests testthat, covr, xml2, spelling, rstudioapi, neonUtilities, rhdf5

Language en-US

NeedsCompilation no

Author Carl Boettiger [aut, cre] (<https://orcid.org/0000-0002-1642-628X>), Quinn Thomas [aut] (<https://orcid.org/0000-0003-1282-7825>), Christine Laney [aut] (<https://orcid.org/0000-0002-4944-2083>), Claire Lunch [aut] (<https://orcid.org/0000-0001-8753-6593>), Noam Ross [ctb] (<https://orcid.org/0000-0002-2136-0000>)

Maintainer Carl Boettiger <cboettig@gmail.com>

Repository CRAN

Date/Publication 2020-10-27 07:50:06 UTC

R topics documented:

`neon_citation` *Generate the appropriate citation for your data*

Description

Generate the appropriate citation for your data

Usage

```
neon_citation(product = NULL, download_date = Sys.Date(), dir = neon_dir())
```

Arguments

`product`	A NEON `productCode`. See neon_download.
`download_date`	Date of download to be included in citation. default is today's date, see details.
`dir`	Location where files should be downloaded. By default will use the appropriate applications directory for your system (see rappdirs::user_data_dir). This default also be configured by setting the environmental variable `NEONSTORE_HOME`, see Sys.setenv or Renviron.

Details

Note that the `neon_download()` does not record download date for each file. Citing a single product download date is after all rather meaningless, as parts of a products may not have all been downloaded on different dates. Indeed, `neon_download()` is designed in precisely this way, to allow easy updating of downloads without re-downloading older data.

Value

returns a utils::bibentry object, which can be used as text or formatted for bibtex.

References

https://www.neonscience.org/data/about-data/data-policies

Examples

```
neon_citation("DP1.10003.001")

## or the citation for all products in store:
neon_citation()

## as bibtex
format(neon_citation("DP1.10003.001"), "bibtex")
```

---

neon_db *Cache-able duckdb database connection*

Description

Cache-able duckdb database connection

Usage

```
neon_db(dir = neon_dir(), ...)
```

Arguments

dir	Location where files should be downloaded. By default will use the appropriate applications directory for your system (see rappdirs::user_data_dir). This default also be configured by setting the environmental variable NEONSTORE_HOME, see Sys.setenv or Renviron.
...	additional arguments to dbConnect

Details

Creates a connection to a permanent duckdb database instance in the provided directory (see neon_dir()). This connection is also cached, so that code which repeatedly calls [neon_db] will not stall or hang.

Examples

```
# tempfile used for illustration only
neon_db(tempfile())
```

`neon_delete_db` *delete the local NEON database*

Description

delete the local NEON database

Usage

```
neon_delete_db(dir = neon_dir(), ask = interactive())
```

Arguments

`dir`	Location where files should be downloaded. By default will use the appropriate applications directory for your system (see rappdirs::user_data_dir). This default also be configured by setting the environmental variable `NEONSTORE_HOME`, see Sys.setenv or Renviron.
`ask`	Ask for confirmation first?

Details

Just a helper function that deletes the NEON database files, which are found under `file.path(neon_dir(),"database")`. This does not delete downloaded raw data, which can easily be re-loaded with `neon_store()`. Usually unnecessary but can be helpful in resetting a corrupt database.

If you want to delete all raw data files downloaded by neonstore as well, simply delete the entire directory given by `neon_dir()`

Examples

```
# Create a db
dir <- tempfile()
neon_db(dir)

# Delete it
neon_delete_db(dir, ask = FALSE)
```

neon_dir *Default directory for persistent NEON file store*

Description

Use `neon_dir()` to view or access the currently active local store. By default, `neon_download()` downloads files into the `neon_dir()`, which uses an appropriate application directory for your operating system, see `rappdirs::user_data_dir()`. This location can be overridden by setting the environmental variable `NEONSTORE_HOME`. `neonstore` functions (e.g. `neon_index()`, and `neon_read()`) look for files in the `neon_dir()` directory by default. (All functions can also take a one-off argument to `dir` in the function call in place of the calling `neon_dir()` to access the default.

Usage

```
neon_dir()
```

Value

the active `neonstore` directory.

Examples

```
neon_dir()

## Override with an environmental variable:
Sys.setenv(NEONSTORE_HOME = tempdir())
neon_dir()
## Unset
Sys.unsetenv("NEONSTORE_HOME")
```

neon_disconnect *Disconnect from the neon database*

Description

Disconnect from the neon database

Usage

```
neon_disconnect(dir = neon_dir())
```

Arguments

dir	Location where files should be downloaded. By default will use the appropriate applications directory for your system (see rappdirs::user_data_dir). This default also be configured by setting the environmental variable NEONSTORE_HOME, see Sys.setenv or Renviron.

---

neon_download *Download NEON data products into a local store*

Description

Download NEON data products into a local store

Usage

```
neon_download(
  product,
  start_date = NA,
  end_date = NA,
  site = NA,
  type = "expanded",
  file_regex = "[.]zip",
  quiet = FALSE,
  verify = TRUE,
  dir = neon_dir(),
  unzip = TRUE,
  api = "https://data.neonscience.org/api/v0",
  .token = Sys.getenv("NEON_TOKEN")
)
```

Arguments

product	A NEON productCode. See neon_download.
start_date	Download only files as recent as (YYYY-MM-DD). Leave as NA to download up to the most recent available data.
end_date	Download only files up to end_date (YYYY-MM-DD). Leave as NA to download all prior data.
site	4-letter site code(s) to filter on. Leave as NA to search all.
type	Should we prefer the basic or expanded version of this product? See details.
file_regex	Download only files matching this pattern. See details.
quiet	Should download progress be displayed?
verify	Should downloaded files be compared against the MD5 hash reported by the NEON API to verify integrity? (default TRUE)

dir	Location where files should be downloaded. By default will use the appropriate applications directory for your system (see rappdirs::user_data_dir). This default also be configured by setting the environmental variable NEONSTORE_HOME, see Sys.setenv or Renviron.
unzip	should we extract .zip files? (default TRUE). Note: .zip files are preserved in the store to avoid repeated downloads.
api	the URL to the NEON API, leave as default.
.token	an authentication token from NEON. A token is not required but will allow access to a higher number of requests before rate limiting applies, see https://data.neonscience.org/data-api/rate-limiting/#api-tokens. Note that once files are downloaded once, neonstore provides persistent access to them without further interaction required with the API.

Details

Each NEON data product consists of a collection of objects (e.g. tables), which are in turn broken into individual files by site and sampling month. Additionally, many NEON products have been expanded, including some additional columns. Consequently, users must specify if they want the "basic" or "expanded" version of this data.

In the products table (see neon_products), the productHasExpanded column indicates if the data product has expanded, and the columns productHasBasicDescription and productHasExpandedDescription provide a detailed explanation of the differences between the "expanded" and "basic" versions of that particular product.

The API provides access to a .zip file containing all the component objects (e.g. tables) for that product at that site and sampling month. Additionally, the API allows users to request component files directly (e.g. as .csv files). Requesting component files directly avoids the additional overhead of downloading other components that are not needed. Both the .zip and relevant .csv and zip files in products that have expanded will include both a "basic" and "expanded" name in the filename. Setting type argument of neon_download() to the preferred one will make it filter out the other one.

By default, neon_download() will request the .zip packet for the product, matching the requested type. neon_download() will extract the component files into the store, removing the .zip file. Specific files within a product can be identified by altering the file_regex argument (see examples).

neon_download() will avoid downloading metadata files which are bitwise identical to other files in the same download request, as indicated by the crc32 hash reported by the API. These typically include metadata that are shared across the product as a whole, but are for some reason included in each sampling month for each site – potentially thousands of duplicates. These duplicates are also packaged within the .zip downloads where it is not possible to exclude them from the download.

Examples

```
neon_download("DP1.10003.001",
              start_date = "2018-01-01",
              end_date = "2019-01-01",
```

```
                site = "YELL")

## Advanced use: filter for a particular table in the product
neon_download(product = "DP1.10003.001",
                start_date = "2018-01-01",
                end_date = "2019-01-01",
                site = "YELL",
                file_regex = ".*brd_countdata.*\\.csv")
```

`neon_download_s3` *Download requested NEON files from an S3 bucket*

Description

Queries the AWS-S3 REST endpoint GET bucket for a file list in (in 1000-file chunks), then filters file names to determine what to download. This should be much faster than the NEON API and avoids rate-limiting.

Usage

```
neon_download_s3(
  product,
  start_date = NA,
  end_date = NA,
  site = NA,
  type = "expanded",
  file_regex = "[.]zip",
  quiet = FALSE,
  verify = TRUE,
  dir = neon_dir(),
  unzip = TRUE,
  api = "https://minio.thelio.carlboettiger.info/neonstore/"
)
```

Arguments

`product`	A NEON `productCode`. See neon_download.
`start_date`	Download only files as recent as (`YYYY-MM-DD`). Leave as `NA` to download up to the most recent available data.
`end_date`	Download only files up to end_date (`YYYY-MM-DD`). Leave as `NA` to download all prior data.
`site`	4-letter site code(s) to filter on. Leave as `NA` to search all.
`type`	Should we prefer the basic or expanded version of this product? See details.
`file_regex`	Download only files matching this pattern. See details.

quiet	Should download progress be displayed?
verify	Should downloaded files be compared against the MD5 hash reported by the NEON API to verify integrity? (default TRUE)
dir	Location where files should be downloaded. By default will use the appropriate applications directory for your system (see rappdirs::user_data_dir). This default also be configured by setting the environmental variable NEONSTORE_HOME, see Sys.setenv or Renviron.
unzip	should we extract .zip files? (default TRUE). Note: .zip files are preserved in the store to avoid repeated downloads.
api	URL to an S3 bucket containing raw NEON data files (in flat file structure like that used by neonstore).

Value

(invisibly) table of requested files and metadata

Examples

```
neon_download("DP1.10003.001",
              start_date = "2018-01-01",
              end_date = "2019-01-01",
              site = "YELL")
```

---

neon_export *export local neon store as a zip archive*

Description

Export all or select files from your neon store as a zip archive. This can be useful if you want to bypass accessing the API, such as for archiving the files required for your analysis so that they can be re-created by other users without an API key, or without waiting for the individual download, or any other tiem you want to share or distribute your local store.

Usage

```
neon_export(
  archive = paste(Sys.Date(), "neonstore.zip", sep = "-"),
  product = NA,
  table = NA,
```

```
  site = NA,
  start_date = NA,
  end_date = NA,
  type = NA,
  ext = NA,
  timestamp = NA,
  hash = NULL,
  dir = neon_dir()
)
```

Arguments

archive	path to the zip archive to be created.#'
product	Include only files matching this NEON productCode(s)
table	Include only files matching this table name (or regex pattern). (optional).
site	4-letter site code(s) to filter on. Leave as NA to search all.
start_date	Download only files as recent as (YYYY-MM-DD). Leave as NA to download up to the most recent available data.
end_date	Download only files up to end_date (YYYY-MM-DD). Leave as NA to download all prior data.
type	Should we prefer the basic or expanded version of this product? See details.
ext	only match files with this file extension(s)
timestamp	only match timestamps prior this. See details in neon_index(). Should be a datetime POSIXct object (or coerce-able string)
hash	name of a hashing algorithm to check file integrity. Can be "md5", "sha1", or "sha256" currently; or set to NULL (default) to skip hash computation.
dir	Location where files should be downloaded. By default will use the appropriate applications directory for your system (see rappdirs::user_data_dir). This default also be configured by setting the environmental variable NEONSTORE_HOME, see Sys.setenv or Renviron.

Value

table of selected files and metadata, from neon_index(), invisibly.

See Also

neon_import(), neon_citation()

Examples

```
archive <- tempfile()
dir <- tempdir()
neon_export(archive, dir = dir)
```

`neon_filename_parser` *NEON filename parser*

Description

Parse filenames into their component metadata. See details for definition of each metadata field, or consult the NEON documentation linked below. https://data.neonscience.org/file-naming-conventions

Usage

```
neon_filename_parser(x)
```

Arguments

x	vector of NEON filenames

Details

Metadata components::

- NEON A four-character alphanumeric code, denoting the organizational origin of the data product and identifying the product as operational; data collected as part of a special data collection exercise are designated by a separate, unique alphanumeric code created by the PI.
- DOM A three-character alphanumeric code, referring to the domain of data acquisition (D01 - D20).
- SITE A four-character alphanumeric code, referring to the site of data acquisition; all sites are designated by a standardized four-character alphabetic code.
- DPL A three-character alphanumeric code, referring to data product processing level.
- PRNUM A five-character numeric code, referring to the data product number (see the Data Product Catalog at http://data.neonscience.org/data-product-catalog).
- REV A three-digit designation, referring to the revision number of the data product. The REV value is incremented by 1 each time a major change is made in instrumentation, data collection protocol, or data processing such that data from the preceding revision is not directly comparable to the new.
- HOR A three-character alphanumeric code for Spatial Index #1. Refers to measurement locations within one horizontal plane. For example, if five surface measurements were taken, one at each of the five soil array plots, the number in the HOR field would range from 001-005.
- VER A three-character alphanumeric code for Spatial Index #2. Refers to measurement locations within one vertical plane. For example, if eight temperature measurements are collected, one at each tower vertical level, the number in the VER field would range from 010-080.
- TMI A three-character alphanumeric code for the Temporal Index. Refers to the temporal representation, averaging period, or coverage of the data product (e.g., minute, hour, month, year, sub-hourly, day, lunar month, single instance, seasonal, annual, multi-annual). 000 = native resolution, 001 = native resolution or 1 minute, 002 = 2 minute, 005 = 5 minute, 015 = 15 minute, 030 = 30 minute, 060 = 60 minutes or 1 hour, 101-103 = native resolution

of replicate sensor 1, 2, and 3 respectively, 999 = Sensor conducts measurements at varied interval depending on air mass.

- DESC An abbreviated description of the data file or table.
- YYYY-MM Represents the year and month of the data in the file.
- PKGTYPE The type of data package downloaded. Options are 'basic', representing the basic download package, or 'expanded', representing the expanded download package (see more information below).
- GENTIME The date-time stamp when the file was generated, in UTC. The format of the date-time stamp is YYYYMMDDTHHmmSSZ.

AOP Products Only (Airborne Observation Platform)::

- FLHTDATE Date of flight, YYYYMMDD
- FLIGHTSTRT Start time of flight, YYYYMMDDHH
- FLHTSTRT Start time of flight, YYMMDDHH
- IMAGEDATETIME Date and time of image capture, YYYYMMDDHHmmSS
- CCCCCC Digital camera serial number
- NNNN Sequential number for indexing files
- NNN Planned flightline number
- R Repeat number
- FFFFFF Numeric code for an individual flightline
- EEEEEE UTM easting of lower left corner
- NNNNNNN UTM northing of lower left corner

Value

a data frame in which filenames have been split into metadata components. Column names indicate the metadata field code, see details section for complete descriptions.

References

https://data.neonscience.org/file-naming-conventions

`neon_import` *Import a previously exported zip archive of raw NEON files*

Description

`neon_import()` only reads in previously saved archives from `neon_export()`. This can be useful in cases where see `neon_download()` to download data directly from NEON.

Usage

```
neon_import(archive, overwrite = TRUE, dir = neon_dir())
```

Arguments

archive	path to the zip archive to be imported
overwrite	should we overwrite any existing files?
dir	Location where files should be downloaded. By default will use the appropriate applications directory for your system (see rappdirs::user_data_dir). This default also be configured by setting the environmental variable NEONSTORE_HOME, see Sys.setenv or Renviron.

See Also

neon_export()

Examples

```
## tempfiles for example purposes only!
archive <- tempfile()
neondir <- tempdir()


neon_export(archive, dir = neondir)
neon_import(archive)
```

---

neon_index *Show information about all files downloaded to the local store*

Description

NEON products consist of several individual components, which are in turn broken up by site and sampling month. By storing these individual files, neonstore enables more reproducible workflows that can be traced back to original, unaltered input data. These atomized files can be quickly and easily combined into unified tables, see neon_read.

Usage

```
neon_index(
  product = NA,
  table = NA,
  site = NA,
  start_date = NA,
  end_date = NA,
  type = NA,
  ext = NA,
  timestamp = NA,
  hash = NULL,
  dir = neon_dir(),
  deprecated = TRUE
)
```

Arguments

product	Include only files matching this NEON productCode(s)
table	Include only files matching this table name (or regex pattern). (optional).
site	4-letter site code(s) to filter on. Leave as NA to search all.
start_date	Download only files as recent as (YYYY-MM-DD). Leave as NA to download up to the most recent available data.
end_date	Download only files up to end_date (YYYY-MM-DD). Leave as NA to download all prior data.
type	Should we prefer the basic or expanded version of this product? See details.
ext	only match files with this file extension(s)
timestamp	only match timestamps prior this. See details in neon_index(). Should be a datetime POSIXct object (or coerce-able string)
hash	name of a hashing algorithm to check file integrity. Can be "md5", "sha1", or "sha256" currently; or set to NULL (default) to skip hash computation.
dir	Location where files should be downloaded. By default will use the appropriate applications directory for your system (see rappdirs::user_data_dir). This default also be configured by setting the environmental variable NEONSTORE_HOME, see Sys.setenv or Renviron.
deprecated	Should the index include files that have since been deprecated by more recent downloads? logical, default TRUE.

Details

File names include metadata such as the file productCode, table name, site, and sampling month, as well as timestamp of creation. neon_index() parses this metadata from the file name string and returns the information in a convenient table, along with a path to each file.

Regarding timestamps: NEON will occasionally publish new versions of previously-released raw data files (which may or may not actually differ). The NEON download API, and hence neon_download(), only serve the most recent of such files, but earlier versions may still exist in your local neonstore if you downloaded them before the updated files were released. By default, neon_read() will always select the most recent of such files, thus avoiding duplication and providing the most updated data. For reproducibility however, it may be necessary to access older version instead. Setting the timestamp argument allows the user to filter out newer files and select the original ones instead. Unfortunately, at this time users cannot request the outdated data files from NEON API. For strict reproducibility, users should also archive their local store.

See Also

neon_download()

Examples

```
neon_index()

## Just bird survey product
neon_index("DP1.10003.001")
```

---

neon_pane *Open NEON database connection pane in RStudio*

Description

This function launches the RStudio "Connection" pane to interactively explore the database.

Usage

```
neon_pane()
```

Examples

```
if (!is.null(getOption("connectionObserver"))) neon_pane()
```

---

neon_products *Table of all NEON Data Products*

Description

Return a table of all NEON Data Products, including product descriptions and the productCode needed for neon_download. (including list-columns).

Usage

```
neon_products(
  api = "https://data.neonscience.org/api/v0",
  .token = Sys.getenv("NEON_TOKEN")
)
```

Arguments

api	the URL to the NEON API, leave as default.
.token	an authentication token from NEON. A token is not required but will allow access to a higher number of requests before rate limiting applies, see https://data.neonscience.org/data-api/rate-limiting/#api-tokens. Note that once files are downloaded once, neonstore provides persistent access to them without further interaction required with the API.

See Also

neon_download

Examples

```
products <- neon_products()

# Or search for a keyword
i <- grepl("bird", products$keywords)
products[i, c("productCode", "productName")]
```

---

neon_read *read in neon tabular data*

Description

read in neon tabular data

Usage

```
neon_read(
  table = NA,
  product = NA,
  site = NA,
  start_date = NA,
  end_date = NA,
  ext = NA,
  timestamp = NA,
  dir = neon_dir(),
  files = NULL,
  sensor_metadata = TRUE,
  altrep = FALSE,
  ...
)
```

Arguments

table	the name of a downloaded NEON table in the store, see neon_index
product	Include only files matching this NEON productCode(s)
site	4-letter site code(s) to filter on. Leave as NA to search all.
start_date	Download only files as recent as (YYYY-MM-DD). Leave as NA to download up to the most recent available data.

end_date	Download only files up to end_date (YYYY-MM-DD). Leave as NA to download all prior data.
ext	only match files with this file extension(s)
timestamp	only match timestamps prior this. See details in neon_index(). Should be a datetime POSIXct object (or coerce-able string)
dir	Location where files should be downloaded. By default will use the appropriate applications directory for your system (see rappdirs::user_data_dir). This default also be configured by setting the environmental variable NEONSTORE_HOME, see Sys.setenv or Renviron.
files	optionally, specify a vector of file paths directly (e.g. as provided from neon_index) and specify table argument as NULL.
sensor_metadata	logical, default TRUE. Should we add metadata fields from file names of sensor data into the table? Adds DomainID, SiteID, horizontalPosition, verticalPosition, and publicationDate. Results in slower parsing.
altrep	enable or disable altrep. Logical, default FALSE. Setting to TRUE can speed up reading, but may cause vroom::vroom to throw mapping error: Too many open files.
...	additional arguments to vroom::vroom, can usually be omitted.

Details

NEON's tabular data files are separated out into separate .csv files for each site for each month of sampling. In principle, each file has identical columns. vroom::vroom can read in a data table that has been sharded into many files like this much much faster than other parsers can read in each table iteratively, (and thus can greatly out-perform the 'stacking" methods in neonUtilities).

When reading in very large numbers of files, it may be helpful to set altrep = FALSE to opt out of vroom's fast altrep mechanism, which can cause neon_read() to fail when stacking thousands of files.

Unfortunately, not all datasets are entirely consistent in their use of columns. neon_read works around this by parsing such tables in groups of matching schema, which is still reasonably fast.

NEON sensor data products currently do not include important metadata columns containing DomainID, SiteID, horizontalPosition, verticalPosition, and publicationDate in the data files themselves, but only encode this in the in the raw file names. All though these values are shared across a raw data file, this information is lost when stacking the tables unless explicit columns are added to the data. This requires us to parse the files one-by-one, which is much slower. By default this information is added to the table, altering the stacked table schema from that of the raw table. Disable this behavior by setting sensor_metadata = FALSE. Future NEON sensor data products may start including this information in the raw data files, as is already the case for observational data.

Examples

```
neon_read("brd_countdata-expanded")

## Sensor inputs will add metadata columns by default
neon_read("waq_instantaneous", site = c("CRAM","SUGG"))
```

neon_sites *Table of all NEON sites*

Description

Returns a table of all NEON sites by making a single API call to the /sites endpoint.

Usage

```
neon_sites(
  api = "https://data.neonscience.org/api/v0",
  .token = Sys.getenv("NEON_TOKEN")
)
```

Arguments

api	the URL to the NEON API, leave as default.
.token	an authentication token from NEON. A token is not required but will allow access to a higher number of requests before rate limiting applies, see https://data.neonscience.org/data-api/rate-limiting/#api-tokens. Note that once files are downloaded once, neonstore provides persistent access to them without further interaction required with the API.

neon_store *import neon data into a local database*

Description

import neon data into a local database

Usage

```
neon_store(
  table = NA,
  product = NA,
  type = NA,
  dir = neon_dir(),
  n = 500L,
  quiet = FALSE,
  ...
)
```

Arguments

table	Include only files matching this table name (or regex pattern). (optional).
product	Include only files matching this NEON productCode(s)
type	Should we prefer the basic or expanded version of this product? See details.
dir	Location where files should be downloaded. By default will use the appropriate applications directory for your system (see rappdirs::user_data_dir). This default also be configured by setting the environmental variable NEONSTORE_HOME, see Sys.setenv or Renviron.
n	number of files that should be read per iteration
quiet	show progress?
...	Arguments passed on to neon_read
	sensor_metadata logical, default TRUE. Should we add metadata fields from file names of sensor data into the table? Adds DomainID, SiteID, horizontalPosition, verticalPosition, and publicationDate. Results in slower parsing.
	altrep enable or disable altrep. Logical, default FALSE. Setting to TRUE can speed up reading, but may cause vroom::vroom to throw mapping error: Too many open files.
	files optionally, specify a vector of file paths directly (e.g. as provided from neon_index) and specify table argument as NULL.
	site 4-letter site code(s) to filter on. Leave as NA to search all.
	start_date Download only files as recent as (YYYY-MM-DD). Leave as NA to download up to the most recent available data.
	end_date Download only files up to end_date (YYYY-MM-DD). Leave as NA to download all prior data.
	ext only match files with this file extension(s)
	timestamp only match timestamps prior this. See details in neon_index(). Should be a datetime POSIXct object (or coerce-able string)

Value

the connection object (invisibly)

---

neon_table *Return a neon table from the database*

Description

Return a neon table from the database

Usage

```
neon_table(table, site = NA, con = neon_db())
```

Arguments

table	the name of a downloaded NEON table in the store, see neon_index
site	4-letter site code(s) to filter on. Leave as NA to search all.
con	a connection to the neon database

Details

We cannot filter on start_date or end_date since these come only from the filename metadata and are only added to instrument tables, not observation tables etc.

Index