Package: cleanNPN (via r-universe)

November 3, 2026

Title Reproducible Workflow for Cleaning National Phenology Network Status and Intensity Data

Version 0.0.0.9000

Description The `cleanNPN` R package provides a reproducible workflow for processing National Phenology Network (NPN) status and intensity phenological data, enabling researchers to quickly clean raw datasets using a suite of built-in functions.

BugReports https://github.com/pitt-songlab/cleanNPN/issues

License file LICENSE

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

Depends R (>= 4.1),

Imports dplyr, lubridate, rnpn, slider, rlang

Suggests knitr, rmarkdown

VignetteBuilder knitr

LazyData true

Config/pak/sysreqs libicu-dev libxml2-dev libssl-dev

Repository https://pitt-songlab.r-universe.dev

Date/Publication 2026-11-02 21:01:12 UTC

RemoteUrl https://github.com/pitt-songlab/cleanNPN

RemoteRef HEAD

RemoteSha 66befcf2fe7262f3ddf6027cc63e530bc1476695

Contents

calc_first_yes 2
clean_by_leading_no 3
clean_by_resolving_conflict 5

clean_by_trailing_yes 7
clean_by_window 8
clean_data 10
download_test_data 13

Index **16**

`calc_first_yes` *Calculate the day of first "yes" observation*

Description

For each combination of individual, phenophase, and calendar year, finds the day of year of the earliest observation in which the phenophase was recorded as present (a "yes"). This is commonly used to estimate the onset date of a phenophase such as first leaf, first flower, or first fruit.

Usage

```
calc_first_yes(data)
```

Arguments

`data`	A data frame or tibble of phenophase observations. Must contain the following columns: **individual_id** Identifier for the individual plant or animal. **phenophase_id** Identifier for the phenophase. **observation_date** Date of the observation. Must be of class `Date` (or otherwise compatible with `lubridate::year()`). `day_of_year` Numeric day of year (1 to 366) of the observation. `phenophase_status` Integer status of the phenophase: `1` = yes, `0` = no, `-1` = uncertain.

Details

Important notes:

- Because the first "yes" is chosen based on day of year within a calendar year, it does not account for phenophases that span the turn of the year.
- Only "yes" records are considered. Records that are "no" or "uncertain", and records with a missing status, are dropped before the first observation is selected. As a result, combinations of individual, phenophase, and year without a "yes" record do not appear in the output.
- The result is unaffected by having multiple "yes" records from different observers for the same individual on the same day.

Value

A tibble with one row per combination of `individual_id`, `phenophase_id`, and `year`, and the following columns:

`phenophase_id` Identifier for the phenophase.

`individual_id` Identifier for the individual plant or animal.

`year` Calendar year of `observation_date`.

`day_of_first_yes` The day of year of the earliest "yes" observation for the same `individual_id` and `phenophase_id` within the calendar year.

Examples

```
obs <- data.frame(
  observation_id = 1:5,
  individual_id = c(101, 101, 101, 101, 102),
  phenophase_id = 373,
  observation_date = as.Date(c(
    "2023-04-10", "2023-04-03", "2023-04-17", "2024-03-28", "2023-04-12"
  )),
  day_of_year = c(100, 93, 107, 88, 102),
  phenophase_status = c(1, 1, 1, 1, 0)
)

calc_first_yes(obs)
```

`clean_by_leading_no` *Count "no" observations in a leading window before each observation*

Description

For each observation, counts how many "no" records were made for the same individual and phenophase during a window of days leading up to, but not including, that observation's date. Conflicting records (records for the same individual and phenophase on the same date that include both a "yes" and a "no") are either included or excluded, as controlled by `include_conflict`.

Usage

```
clean_by_leading_no(data, leading_window_days = 7, include_conflict = TRUE)
```

Arguments

`data`	A data frame or tibble of phenophase observations. Must contain the following columns: `individual_id` Identifier for the individual plant or animal. `phenophase_id` Identifier for the phenophase. `observation_date` Date of the observation. Must be of class `Date` .

`phenophase_status` Integer status of the phenophase: `1` = yes, `0` = no, `-1` = uncertain.

`leading_window_days`
: A single positive number giving the length of the look-back window, in days. The window starts this many days before each observation date and ends the day before it. Defaults to `7`.

`include_conflict`
: A single logical value specifying whether to include conflicting "no" records when counting leading "no" records. Defaults to `TRUE`.

Details

Important notes:

- Observations are ordered by `observation_date`, **not** by `day_of_year`. Because the window is defined using calendar dates, the count can cross over January 1st. For example, with `leading_window_days = 7`, an observation on January 3 will count "no" records from the previous December, and so on across any year boundary.
- `leading_no_count` counts the number of "no" **records** . in the window, not the number of consecutive "no" days. Days without any . observation do not contribute to the count. .
- This function is the backward-looking counterpart to `clean_by_trailing_yes()`, which counts "yes" records in the window **after** each observation.

Value

A tibble containing all rows of `data`, sorted by `individual_id`, `phenophase_id`, and `observation_date`, with all original columns plus two new columns:

`leading_no_count` The number of "no" records for the same `individual_id` and `phenophase_id` within the leading window before each observation date (excluding the date itself).

`is_conflict` Logical. `TRUE` if the record's individual, phenophase, and date have both a "yes" and a "no" record.

Examples

```
obs <- data.frame(
  individual_id = 1,
  phenophase_id = 501,
  observation_date = as.Date(c(
    "2023-12-28", "2023-12-31", "2024-01-02",
    "2024-01-03", "2024-01-03"
  )),
  phenophase_status = c(0, 0, 0, 1, 0)
)

# The two records on Jan 3 conflict (one "yes" and one "no"). Counts for
# early January include "no" records from December of the prior year.
clean_by_leading_no(obs, leading_window_days = 7, include_conflict = TRUE)

# Exclude conflicting "no" records, which lowers the count for Jan 5
```

```
clean_by_leading_no(obs, leading_window_days = 7, include_conflict = FALSE)
```

`clean_by_resolving_conflict` *Resolve conflicting observations by keeping the most experienced observer*

Description

Cleans phenophase observation data in which multiple observers have reported on the same individual, phenophase, and date. When observers disagree or duplicate each other, the record from the observer with the most total observations (a proxy for experience) is retained. Uncertain and missing status records are removed.

Usage

```
clean_by_resolving_conflict(
  data,
  calculate_observation_option = c("by_year", "all_years", "other"),
  calculate_observation_by = NULL
)
```

Arguments

`data` A data frame or tibble of phenophase observations. Must contain the following columns, plus any columns named in `calculate_observation_by`:

- `individual_id` Identifier for the individual plant or animal.
- `phenophase_id` Identifier for the phenophase.
- `observation_date` Date of the observation. Must be of class `Date`.
- `observedby_person_id` Identifier for the observer.
- `phenophase_status` Integer status: `1` = yes, `0` = no, `-1` = uncertain.

`calculate_observation_option`
A character string specifying how each observer's number of observations is calculated. One of `"by_year"` (the default), `"all_years"`, or `"other"`. If `"other"`, `calculate_observation_by` must also be supplied. See Details.

`calculate_observation_by`
A character vector of column name(s) in `data` used to group observations when `calculate_observation_option = "other"`. Required for `"other"`; an error is thrown if it is `NULL`. Ignored for the other options. Defaults to `NULL`.

Details

The function performs the following steps:

1. Counts how many observations each observer has made, using the grouping set by `calculate_observation_option`.

2. Removes observations with an uncertain or missing status.
3. When multiple observers recorded the same individual, phenophase, and date, keeps the record from the observer with the most observations.

Choosing how observations are counted. The `calculate_observation_option` argument controls the groups over which each observer's observations are tallied:

`"by_year"` Counts per observer per year (`observedby_person_id`, `year`).

`"all_years"` Counts per observer across the entire dataset (`observedby_person_id` only).

`"other"` Counts per observer within the additional grouping column(s) given in `calculate_observation_by` (e.g., a site column). `calculate_observation_by` is required for this option; an error is thrown if it is `NULL`. `observedby_person_id` is always included automatically. The derived `year` column may be used here.

Important notes:

- Observations are counted *before* uncertain (`-1`) and `NA` records are removed, so those records contribute to an observer's total.
- Ties are preserved (`with_ties = TRUE`). If two observers have the same `num_of_observations` for a given individual, phenophase, and date, both records are kept, so the output may still contain more than one record per date.
- Records with no conflict (only one observer for that individual, phenophase, and date) are always retained, provided their status is known.

Value

A tibble containing the filtered observations with all original columns and a new column:

`num_of_observations` Number of observations attributed to the record's observer within the grouping defined by `calculate_observation_option`.

`year` Calendar year of `observation_date`. Overwrites any existing `year` column in `data`.

Examples

```
obs <- data.frame(
  individual_id = 1,
  phenophase_id = 501,
  observation_date = as.Date(c(
    "2024-04-01", "2024-04-01", "2024-04-02", "2024-04-03", "2024-04-03"
  )),
  observedby_person_id = c(10, 20, 10, 10, 20),
  site_id = "A",
  phenophase_status = c(1, 0, 1, 1, 0)
)

# Default: count observations per observer per year
clean_by_resolving_conflict(obs)

# Count observations per observer across all years
clean_by_resolving_conflict(obs, calculate_observation_option = "all_years")
```

```
# Count observations per observer within each site
clean_by_resolving_conflict(
  obs,
  calculate_observation_option = "other",
  calculate_observation_by = "site_id"
)

# "other" requires calculate_observation_by, so this raises an error
try(clean_by_resolving_conflict(obs, calculate_observation_option = "other"))
```

`clean_by_trailing_yes` *Count "yes" observations in a trailing window after each observation*

Description

For each observation, counts how many "yes" records were made for the same individual and phenophase during a window of days following, but not including, that observation's date. Conflicting records (records for the same individual and phenophase on the same date that include both a "yes" and a "no") are either included or excluded, as controlled by `include_conflict`.

Usage

```
clean_by_trailing_yes(data, trailing_window_days = 7, include_conflict = TRUE)
```

Arguments

`data` A data frame or tibble of phenophase observations. Must contain the following columns:

- `individual_id` Identifier for the individual plant or animal.
- `phenophase_id` Identifier for the phenophase.
- `observation_date` Date of the observation. Must be of class `Date`.
- `phenophase_status` Integer status of the phenophase: `1` = yes, `0` = no, `-1` = uncertain.

`trailing_window_days`
: A single postive number giving the length of the forward-looking window, in days. The window starts the day after each observation date and ends this many days after it. Defaults to `7`.

`include_conflict`
: A single logical value specifying whether to include conflicting "yes" records when counting trailing "yes" records. Defaults to `TRUE`.

Details

Important notes:

- Observations are ordered by `observation_date`, **not** by `day_of_year`. Because the window is defined using calendar dates, the count can cross over January 1st. For example, with `trailing_window_days = 7`, an observation on December 29 will count "yes" records from the following January, and so on across any year boundary.
- `trailing_yes_count` counts the number of "yes" **records** in the window, not the number of consecutive "yes" days. Days without any observation do not contribute to the count.
- This function is the forward-looking counterpart to `clean_by_leading_no()`, which counts "no" records in the window **before** each observation.

Value

A tibble containing all rows of `data`, sorted by `individual_id`, `phenophase_id`, and `observation_date`, with all original columns plus two new columns:

`trailing_yes_count` The number of "yes" records for the same `individual_id` and `phenophase_id` within the trailing window after each observation date (excluding the date itself).

`is_conflict` Logical. `TRUE` if the record's individual, phenophase, and date have both a "yes" and a "no" record.

Examples

```
obs <- data.frame(
  individual_id = 1,
  phenophase_id = 501,
  observation_date = as.Date(c(
    "2023-12-27", "2023-12-29", "2024-01-02",
    "2024-01-03", "2024-01-03"
  )),
  phenophase_status = c(0, 0, 1, 1, 0)
)

# The two records on Jan 3 conflict (one "yes" and one "no"). Counts for
# late December include "yes" records from January of the following year.
clean_by_trailing_yes(obs, trailing_window_days = 7, include_conflict = TRUE)

# Exclude conflicting "yes" records
clean_by_trailing_yes(obs, trailing_window_days = 7, include_conflict = FALSE)
```

`clean_by_window` *Restrict observations to a day-of-year window*

Description

Keeps only the observations whose day_of_year falls within a specified range. This is useful for excluding records outside the part of the year in which a phenophase is biologically plausible (for example, removing leaf-out records from late autumn). Either bound may be left open.

Usage

```
clean_by_window(data, min_doy = NULL, max_doy = NULL)
```

Arguments

data	A data frame or tibble of phenophase observations. Must contain the following column: day_of_year Numeric day of year (1 to 366) of the observation.
min_doy	A single number giving the earliest day of year (inclusive) to keep. If NULL (the default), there is no lower limit.
max_doy	A single number giving the latest day of year (inclusive) to keep. If NULL (the default), there is no upper limit.

Details

Both bounds are inclusive: a record is kept if day_of_year >= min_doy and day_of_year <= max_doy. If min_doy is NULL, no lower limit is applied, and if max_doy is NULL, no upper limit is applied. If both are NULL, all records with a non-missing day_of_year are returned.

Important notes:

- Records with NA in day_of_year are always removed. This happens even when both bounds are NULL.
- The window is a simple range on a single calendar year's day numbering, so it cannot represent a window that wraps around the turn of the year (e.g., day 330 to day 30). If min_doy > max_doy, no rows are returned.
- Day of year runs from 1 to 366, with 366 occurring only in leap years. Because the same day of year corresponds to different calendar dates in leap and non-leap years, dates after February can differ by one day between years.

Value

A tibble containing the filtered observations with all original columns and only the rows with day_of_year between min_doy and max_doy.

Examples

```
obs <- data.frame(
  individual_id = 1,
  phenophase_id = 501,
  day_of_year = c(10, 60, 100, 150, 200, 300),
  phenophase_status = c(0, 0, 1, 1, 1, 0)
)
```

```
# Keep days 60 to 200 (inclusive)
clean_by_window(obs, min_doy = 60, max_doy = 200)

# Only a lower limit
clean_by_window(obs, min_doy = 100)

# Only an upper limit
clean_by_window(obs, max_doy = 100)

# Records with NA in day_of_year are always dropped
obs_na <- data.frame(
  individual_id = 1,
  phenophase_id = 501,
  day_of_year = c(10, NA, 100, NA, 200, 300),
  phenophase_status = c(0, 1, 1, 0, 1, 0)
)

# The two NA rows are removed along with the days outside the window
clean_by_window(obs_na, min_doy = 60, max_doy = 200)

# NA rows are removed even when both bounds are NULL
clean_by_window(obs_na)
```

`clean_data` *Clean phenophase observations using a configurable sequence of filters*

Description

Applies up to four cleaning steps to USA-NPN status and intensity observations:

1. Resolve conflicting reports from different observers for the same individual, phenophase, and date, keeping the record from the observer with the most observations.
2. Count the number of leading "no" observations before each record, and filter on a specified threshold.
3. Count the number of trailing "yes" observations after each record, and filter on a specified threshold.
4. Filter records to those within the specified day-of-year window.

Each step is optional and runs only when its arguments are supplied, so you can combine them as needed. Rows that fail the thresholds you set are removed from the output.

Usage

```
clean_data(
  data,
  calculate_observation_option = NULL,
```

```
  calculate_observation_by = NULL,
  min_leading_no = NULL,
  leading_window_days = NULL,
  min_trailing_yes = NULL,
  trailing_window_days = NULL,
  include_conflict = TRUE,
  min_doy = NULL,
  max_doy = NULL
)
```

Arguments

`data`	A data frame or tibble of phenophase observations. Must contain the following columns: `individual_id` Identifier for the individual plant or animal. `phenophase_id` Identifier for the phenophase. `observation_date` Date of the observation. Must be of class `Date`. `phenophase_status` Integer status of the phenophase: `1` = yes, `0` = no, `-1` = uncertain. `observedby_person_id` Identifier for the observer. Required when resolving conflicts. `day_of_year` Numeric day of year (1 to 366) of the observation. Required when `min_doy` or `max_doy` is supplied. Additional columns named in `calculate_observation_by` are also required when `calculate_observation_option = "other"`.
`calculate_observation_option`	A character string, one of `"by_year"`, `"all_years"`, or `"other"`, specifying how each observer's number of observations is counted when resolving conflicts. See `clean_by_resolving_conflict()` for details. Defaults to `NULL` (step skipped).
`calculate_observation_by`	A character vector of column name(s) used to group observations when `calculate_observation_option = "other"`. See `clean_by_resolving_conflict()` for details. Defaults to `NULL`.
`min_leading_no`	A single number giving the minimum `leading_no_count` a record must have to be kept. Must be supplied together with `leading_window_days`. See `clean_by_leading_no()` for details. Defaults to `NULL` (step skipped).
`leading_window_days`	A single positive number giving the length, in days, of the window preceding each observation in which "no" records are counted. Must be supplied together with `min_leading_no`; if only one of the two is supplied, the step is skipped. See `clean_by_leading_no()` for details. Defaults to `NULL` (step skipped).
`min_trailing_yes`	A single number giving the minimum `trailing_yes_count` a record must have to be kept. Must be supplied together with `trailing_window_days`; if only one of the two is supplied, the step is skipped. See `clean_by_trailing_yes()` for details. Defaults to `NULL` (step skipped).

`trailing_window_days`
: A single positive number giving the length, in days, of the window following each observation in which "yes" records are counted. Must be supplied together with `min_trailing_yes`; if only one of the two is supplied, the step is skipped. See `clean_by_trailing_yes()` for details. Defaults to `NULL` (step skipped).

`include_conflict`
: A single logical value specifying whether to count same-date conflicting "yes" and "no" records in the leading and trailing window calculations. See `clean_by_leading_no()` and `clean_by_trailing_yes()` for details. Defaults to `TRUE`. Ignored unless the leading or trailing step runs.

`min_doy`, `max_doy`
: Single numbers giving the lower and upper bounds (inclusive) of the day of year, from 1 to 366, of the observations to retain. See `clean_by_window()` for details. Either may be `NULL` to leave that bound open. If both are `NULL` (the default), the window step is skipped.

Details

Important notes:

- Both the leading and trailing counts are calculated after conflict resolution, so the impact of `include_conflict` is minimal if you have already resolved conflicts.
- The leading and trailing counts are both calculated before either threshold is applied, so the two thresholds do not affect each other's counts.
- Progress messages are printed for each step that runs. Use `suppressMessages()` to silence them.

Value

A tibble of the retained observations. If no step runs, `data` is returned unchanged. Depending on which steps ran, the result contains all original columns and some of:

`num_of_observations` Added when conflicts are resolved.

`year` Added when conflicts are resolved.

`leading_no_count` Added when the leading "no" step runs.

`trailing_yes_count` Added when the trailing "yes" step runs.

`is_conflict` Added when the leading "no" or trailing "yes" step runs. Indicates whether the record's individual, phenophase, and date have both a "yes" and a "no" record.

Examples

```
obs <- data.frame(
  individual_id = 1,
  phenophase_id = 501,
  observation_date = as.Date("2024-04-01") + c(0, 2, 2, 4, 6, 8, 10),
  observedby_person_id = c(10, 10, 20, 10, 10, 10, 10),
  phenophase_status = c(0, 0, 1, 1, 1, 1, 0)
)
```

```
# The day-of-year window step requires a `day_of_year` column
obs$day_of_year <- as.integer(format(obs$observation_date, "%j"))

# Resolve observer conflicts only
clean_data(obs, calculate_observation_option = "by_year")

# Keep records followed by at least 2 "yes" observations within 7 days
clean_data(obs, min_trailing_yes = 2, trailing_window_days = 7)

# Include same-date conflicts in the count
clean_data(
  obs,
  min_trailing_yes = 2, trailing_window_days = 7,
  include_conflict = TRUE
)

# Exclude same-date conflicts from the count
clean_data(
  obs,
  min_trailing_yes = 2, trailing_window_days = 7,
  include_conflict = FALSE
)

# Restrict to a day_of_year window (uses the `day_of_year` column)
clean_data(obs, min_doy = 95, max_doy = 100)

# Combine several steps
clean_data(
  obs,
  calculate_observation_option = "all_years",
  min_leading_no = 1, leading_window_days = 7,
  min_trailing_yes = 1, trailing_window_days = 7,
  min_doy = 60, max_doy = 200
)
```

`download_test_data` *Download example phenology data for red maple from the USA-NPN*

Description

Downloads one year (2025) of individual phenophase status and intensity records for red maple (*Acer rubrum*) from the USA National Phenology Network (USA-NPN) using the **rnpn** package. It is intended for generating example data to demonstrate and test the cleaning functions in this package.

Usage

```
download_test_data(phenophase = c("leaf", "flower"))
```

Arguments

phenophase	A character string specifying which phenophase class to download. One of "leaf" (the default) or "flower".

Details

The query is fixed to the following parameters, so the only choice you make is the phenophase:

- **Species:** red maple (USA-NPN species ID 3).
- **Year:** 2025.
- **Additional fields:** ObservedBy_Person_ID, Site_Name, and Dataset_ID, requested in addition to the default columns returned by rnpn::npn_download_status_data().
- **Request source:** "package_example_dataset_cleanNPN", which identifies the download to the USA-NPN as coming from this package.

The phenophase argument selects the USA-NPN phenophase class:

"leaf" Phenophase class ID 3.

"flower" Phenophase class ID 7.

Important notes:

- An internet connection is required, and the download may take some time depending on the size of the request and the USA-NPN server load.
- The USA-NPN database is updated continuously, so repeated calls may return different records (for example, late-submitted or revised observations).

Value

A data frame of status-and-intensity observations as returned by rnpn::npn_download_status_data(), with one row per observation record. It includes the standard USA-NPN columns (such as individual_id, phenophase_id, observation_date, day_of_year, and phenophase_status) plus observedby_person_id, site_name, and dataset_id.

Examples

```
## Not run:
# Requires an internet connection
leaf_obs <- download_test_data("leaf")
head(leaf_obs)

flower_obs <- download_test_data("flower")

# Use the downloaded data with the cleaning functions
cleaned <- clean_data(
  leaf_obs,
  calculate_observation_option = "by_year",
  min_trailing_yes = 1, trailing_window_days = 7
)
```

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

Index

calc_first_yes, 2
clean_by_leading_no, 3
clean_by_leading_no(), *8*, *11*, *12*
clean_by_resolving_conflict, 5
clean_by_resolving_conflict(), *11*
clean_by_trailing_yes, 7
clean_by_trailing_yes(), *4*, *11*, *12*
clean_by_window, 8
clean_by_window(), *12*
clean_data, 10

download_test_data, 13

lubridate::year(), *2*

rnpn::npn_download_status_data(), *14*

suppressMessages(), *12*