Package ‘pep725’

March 31, 2026

Title Pan-European Phenological Data Analysis

Version 1.0.2

Description Provides a framework for quality-aware analysis of ground-based phenological data from the PEP725 Pan-European Phenology Database (Templ et al. (2018) <doi:10.1007/s00484-018-1512-8>; Templ et al. (2026) <doi:10.1111/nph.70869>) and similar observation networks. Implements station-level data quality grading, outlier detection, phenological normals (climate baselines), anomaly detection, elevation and latitude gradient estimation with robust regression, spatial synchrony quantification, partial least squares (PLS) regression for identifying temperature-sensitive periods, and sequential Mann-Kendall trend analysis. Supports data import from PEP725 files, conversion of user-supplied data, and downloadable synthetic datasets for teaching without barriers of registration.
All analysis outputs provide 'print', 'summary', and 'plot' methods. Interactive spatial visualization is available via 'leaflet'.

License GPL-3

Encoding UTF-8

LazyData true

Depends R (>= 4.1), data.table, ggplot2

Imports dplyr, mgcv, methods, patchwork, purrr, robustbase, stats, tibble, tidyr, tidyselect, utils, rnaturalearth, sf

Suggests ggmap, knitr, leaflet, miniUI, nlme, quantreg, rmarkdown, rnaturalearthdata, pls, shiny, sp, testthat (>= 3.0.0)

Config/testthat/edition 3

VignetteBuilder knitr

RoxygenNote 7.3.3

BugReports `https://github.com/matthias-da/pep725/issues`

URL `https://github.com/matthias-da/pep725`

NeedsCompilation no

Author Matthias Templ [aut, cre] (ORCID:
<https://orcid.org/0000-0002-8638-5276>),
Barbara Templ [aut] (ORCID: <https://orcid.org/0000-0002-9391-5888>)

Maintainer Matthias Templ <matthias.templ@gmail.com>

Repository CRAN

Date/Publication 2026-03-31 10:50:02 UTC

Contents

add_country 4
add_daylength 5
as.pep 5
bbch_description 6
calc_daylength 7
calc_max_daylength 8
calc_thermal_sum 9
calc_thermal_units 10
get_phenology_doys 12
is.pep 14
kendall_tau 15
new_pep 16
pep-class 17
pep725_demo 17
pep_cache_clear 18
pep_cache_info 19
pep_check_connectivity 20
pep_check_phases 21
pep_check_phases_multi 23
pep_completeness 24
pep_coverage 26
pep_download 28
pep_flag_outliers 29
pep_import 31
pep_plot_outliers 33
pep_quality 35
pep_second_events 37
pep_seed 40
pep_simulate 41
pheno_anomaly 42
pheno_combine 45
pheno_gradient 48
pheno_leaflet 50
pheno_map 52
pheno_normals 54
pheno_plot 57
pheno_plot_hh 58
pheno_plot_timeseries 59

pheno_pls . . . 61
pheno_regional . . . 63
pheno_regional_hh . . . 65
pheno_synchrony . . . 67
pheno_trend_turning . . . 69
plot.pep . . . 71
plot.pep_completeness . . . 71
plot.pep_coverage . . . 72
plot.pep_outliers . . . 73
plot.pep_quality . . . 73
plot.pheno_anomaly . . . 75
plot.pheno_combined . . . 76
plot.pheno_gradient . . . 77
plot.pheno_normals . . . 77
plot.pheno_pls . . . 78
plot.pheno_synchrony . . . 79
plot.pheno_turning . . . 79
plot.second_events . . . 80
plot_phenology_trends . . . 81
print.pep . . . 83
print.pep_completeness . . . 84
print.pep_coverage . . . 84
print.pep_outliers . . . 85
print.pep_quality . . . 85
print.phase_check . . . 86
print.phase_check_multi . . . 86
print.pheno_anomaly . . . 87
print.pheno_combined . . . 88
print.pheno_gradient . . . 88
print.pheno_normals . . . 89
print.pheno_pls . . . 89
print.pheno_synchrony . . . 90
print.pheno_turning . . . 90
print.second_events . . . 91
select_phase . . . 91
subspecies_report . . . 92
summarize_subspecies_availability . . . 94
summary.pep . . . 96
summary.pep_completeness . . . 97
summary.pep_outliers . . . 97
summary.pep_quality . . . 98
summary.pheno_anomaly . . . 98
summary.pheno_normals . . . 99
summary.pheno_pls . . . 100
summary.pheno_synchrony . . . 100
summary.second_events . . . 101
[.pep . . . 101

Index **102**

`add_country` *Add Country Information to a Dataset Based on Latitude/Longitude*

Description

This function adds a new column `country` by performing a fast spatial join between point coordinates (`lat`, `lon`) and world country polygons from **rnaturalearth**. Geometry is handled via **sf**.

Usage

```
add_country(dt, keep_geometry = FALSE)
```

Arguments

`dt`	A `data.frame` or `data.table` containing columns `lat` and `lon`.
`keep_geometry`	Logical. If `TRUE`, the sf geometry column is kept. Default is `FALSE`.

Details

The input data must contain numeric columns named `lat` and `lon` in WGS84 (EPSG:4326). Rows with `NA` coordinates receive `country` = `NA`.

After assigning countries, Germany is automatically split into `"Germany-North"` and `"Germany-South"` using a latitude threshold:

- lat >= 50 $\rightarrow$ `"Germany-North"`
- lat < 50 $\rightarrow$ `"Germany-South"`

If the input already contains a `country` column it is removed and replaced by the spatially derived one.

Value

A `data.table` with a new column `country`.

Author(s)

Matthias Templ

Examples

```
small <- data.table(
  lat = c(48.2, 51.0, 47.5),
  lon = c(16.3, 10.1, 8.5)
)

pep_with_country <- add_country(small)
pep_with_country
```

`add_daylength` *Add Daylength to Phenological Data*

Description

Adds a daylength column to a phenological dataset based on the observation day and station latitude.

Usage

```
add_daylength(pep)
```

Arguments

`pep`	A `pep` object or data.table with `day` and `lat` columns.

Value

The input data with an added `daylength` column (hours).

Author(s)

Matthias Templ

See Also

`calc_daylength` for the underlying calculation

Examples

```
pep <- pep_download()
pep <- add_daylength(pep)
head(pep[, .(day, lat, daylength)])
```

`as.pep` *Coerce to PEP725 Data Object*

Description

Coerce to PEP725 Data Object

Usage

```
as.pep(x, ...)
```

Arguments

`x`	A data.frame or data.table to coerce.
`...`	Additional arguments (currently unused).

Value

An object of class `pep`.

Author(s)

Matthias Templ

`bbch_description` *Get BBCH Phase Description*

Description

Returns human-readable descriptions for BBCH phenological codes.

Usage

```
bbch_description(codes, na.rm = TRUE, sort = TRUE)
```

Arguments

`codes`	Integer vector of BBCH codes.
`na.rm`	Logical. If `TRUE` (default), removes NA codes and codes without a known description from the output.
`sort`	Logical. If `TRUE` (default), sorts the output by phase code.

Value

A data.frame with columns `phase_id` and `description`, ordered by phase code (if `sort` = `TRUE`).

Author(s)

Matthias Templ

Examples

```
bbch_description(c(60, 65, 100))
bbch_description(c(100, 60, 65, NA), na.rm = TRUE, sort = TRUE)
```

`calc_daylength` *Calculate Daylength (Photoperiod)*

Description

Computes the astronomical daylength (hours of daylight) for a given day of year and latitude. Photoperiod is an important environmental driver of plant phenology.

Usage

```
calc_daylength(doy, lat)
```

Arguments

`doy`	Integer or numeric vector. Day of year (1-365/366).
`lat`	Numeric. Latitude in decimal degrees. Positive for Northern Hemisphere, negative for Southern Hemisphere.

Details

The calculation uses standard astronomical formulas based on the solar declination angle. The formula assumes a "flat horizon" and does not account for atmospheric refraction, elevation, or local topography.

At polar latitudes (|lat| > 66.5), continuous daylight or darkness may occur around solstices.

Value

A list with components:

daylength Daylength in hours

declination Solar declination angle in degrees

If `doy` is a vector, returns a data.frame with these columns.

Phenological Relevance

Many phenological events are triggered by photoperiod thresholds:

- Spring bud burst often requires both temperature accumulation and minimum daylength
- Autumn leaf senescence may be triggered by shortening days
- Flowering in many species is photoperiod-dependent

Author(s)

Matthias Templ

References

Forsythe, W. C., Rykiel, E. J., Stahl, R. S., Wu, H., and Schoolfield, R. M. (1995). A model comparison for daylength as a function of latitude and day of year. *Ecological Modelling*, 80(1), 87–95. doi:10.1016/03043800(94)00034F

Examples

```
# Daylength at spring equinox (DOY 80) at 50°N
calc_daylength(80, 50)

# Summer solstice at different latitudes
calc_daylength(172, c(30, 45, 60))

# Daylength through the year at 45°N
yearly <- calc_daylength(1:365, 45)
plot(yearly$doy, yearly$daylength, type = "l",
     xlab = "Day of Year", ylab = "Daylength (hours)")
```

`calc_max_daylength` *Maximum Daylength at a Latitude*

Description

Calculates the maximum possible daylength (at summer solstice) for a given latitude.

Usage

```
calc_max_daylength(lat)
```

Arguments

`lat`	Numeric. Latitude in decimal degrees.

Value

Numeric. Maximum daylength in hours.

Author(s)

Matthias Templ

Examples

```
# Maximum daylength at different latitudes
calc_max_daylength(c(0, 30, 45, 60, 66.5))
```

`calc_thermal_sum` *Calculate Thermal Sum at Phenological Events*

Description

Computes the accumulated thermal units (GDD) from a start date to the observed phenological event date. Useful for determining the thermal requirements of different phenophases.

Usage

```
calc_thermal_sum(
  pep,
  temp_data,
  t_base = 5,
  t_start = 1,
  by = NULL,
  method = "average"
)
```

Arguments

`pep`	A `pep` object or data.table with phenological observations. Must contain `year` and `day` columns.
`temp_data`	A data.frame or data.table with daily temperature data. Must contain columns: `year`, `doy` (day of year), `tmin`, `tmax` (or `tmean`).
`t_base`	Numeric. Base temperature (°C). Default 5.
`t_start`	Integer. Start day of year for accumulation. Default 1 (Jan 1). Use 60 for March 1, etc.
`by`	Character vector. Columns to match temperature data to phenology (e.g., `"s_id"` for station-specific temperatures). Default `NULL` uses same temperature for all observations in a year.
`method`	Character. GDD calculation method (see `calc_thermal_units`).

Details

This function joins phenological observations with temperature data and calculates the thermal sum (accumulated GDD) at each observation date. This is useful for:

- Determining thermal requirements of phenophases
- Comparing thermal sums across years, locations, or species
- Validating phenological models

Value

The input data with an additional `thermal_sum` column containing the accumulated GDD from `t_start` to the observed DOY.

Author(s)

Matthias Templ

See Also

`calc_thermal_units` for the underlying GDD calculation

Examples

```
# Load phenology data
data(pep_seed)

# Create example temperature data (normally from weather stations/reanalysis)
temp <- data.frame(
  year = rep(2000:2015, each = 365),
  doy = rep(1:365, 16),
  tmin = rnorm(365 * 16, mean = 5, sd = 8),
  tmax = rnorm(365 * 16, mean = 15, sd = 8)
)

# Add thermal sum to phenology data
pep_thermal <- calc_thermal_sum(pep_seed, temp, t_base = 5)
```

calc_thermal_units *Calculate Thermal Units (Growing Degree Days)*

Description

Computes accumulated thermal units (growing degree days, GDD) from daily temperature data. Thermal time is a fundamental concept in phenology, representing the heat accumulation that drives plant development.

Usage

```
calc_thermal_units(
  tmin,
  tmax = NULL,
  t_base = 5,
  t_cap = NULL,
  method = c("average", "modified", "single_sine"),
  cumulative = TRUE,
  na.rm = TRUE
)
```

Arguments

`tmin`	Numeric vector of daily minimum temperatures (°C).
`tmax`	Numeric vector of daily maximum temperatures (°C). If `NULL`, `tmin` is treated as daily mean temperature.
`t_base`	Numeric. Base temperature (°C) below which no development occurs. Default is 5°C, typical for temperate crops.
`t_cap`	Numeric. Optional upper temperature cap (°C). Temperatures above this are set to `t_cap` before GDD calculation. Default `NULL` (no cap). Use ~30°C for crops where high heat doesn't increase development.
`method`	Character. Calculation method: **"average"** (Default) GDD = max(0, (tmax + tmin)/2 - t_base) **"modified"** Sets negative daily GDD to 0 before averaging **"single_sine"** Sine wave approximation for more accurate heat units
`cumulative`	Logical. If `TRUE` (default), returns cumulative sum. If `FALSE`, returns daily values.
`na.rm`	Logical. Remove NA values in accumulation? Default `TRUE`.

Details

Growing Degree Days (GDD) quantify heat accumulation above a base temperature. Plants require a certain thermal sum to reach phenological stages (e.g., flowering typically requires 500-1500 GDD depending on species).

The base temperature represents the minimum temperature for growth:

- Cool-season crops (wheat, barley): 0-5°C
- Warm-season crops (maize, soybean): 10°C
- Fruit trees (apple, cherry): 4-7°C
- Grapevine: 10°C

Value

Numeric vector of thermal units (degree-days). Same length as input. If `cumulative = TRUE`, returns running sum.

Methods

average Simple average: GDD = max(0, (Tmax + Tmin)/2 - Tbase). Most common method, used by many agricultural agencies.

modified Like average, but Tmin and Tmax are first bounded to the range from Tbase to Tcap before averaging. Prevents negative contributions.

single_sine Approximates the daily temperature curve as a sine wave, computing the area above Tbase. More accurate when temperatures cross the base threshold during the day.

Author(s)

Matthias Templ

References

McMaster, G.S., Wilhelm, W.W. (1997). Growing degree-days: one equation, two interpretations. Agricultural and Forest Meteorology 87:291-300.

See Also

`calc_thermal_sum` for computing thermal sum at phenological observations

Examples

```
# Daily temperatures for a week
tmin <- c(5, 7, 8, 10, 12, 11, 9)
tmax <- c(15, 18, 20, 22, 25, 23, 19)

# Cumulative GDD with base 10°C
calc_thermal_units(tmin, tmax, t_base = 10)

# Daily GDD values
calc_thermal_units(tmin, tmax, t_base = 10, cumulative = FALSE)

# With upper cap at 30°C
calc_thermal_units(tmin, tmax, t_base = 10, t_cap = 30)

# Using mean temperature only
tmean <- (tmin + tmax) / 2
calc_thermal_units(tmean, t_base = 10)
```

---

get_phenology_doys *Compute CTRL and SCEN Phenology Day-of-Year Values*

Description

This function computes the mean phenological day-of-year (DOY) for two phenophases (e.g. BBCH 65 and 87) for each country, using either a simple warming-shift model or a GDD-based model.

Usage

```
get_phenology_doys(
  pep,
  phase1,
  phase2,
  genus_name,
  phenology_method = c("robust_shift", "simple_shift", "gdd_cmip6"),
  ctrl_years = 2011:2021,
  scen_years = 2085:2095,
  calib_years = 1991:2020,
  scen_warming = 3,
  shift_per_deg = -4,
  cmip6_tas = NULL,
  gdd_base = 5,
  gdd_threshold1 = 150,
  gdd_threshold2 = 350,
  regions = NULL,
  delta_Tgl = 2.09,
  giss_data = NULL
)
```

Arguments

pep	A data frame containing PEP725 phenology observations, including columns `country`, `day`, `year`, `phase_id`, and `genus`.
phase1	Integer. First phenophase (e.g. 65).
phase2	Integer. Second phenophase (e.g. 87).
genus_name	Character. Genus to filter (e.g. "Malus").
phenology_method	Either `"simple_shift"` or `"gdd_cmip6"`.
ctrl_years	Integer vector. Years for CTRL period (default 2011:2021).
scen_years	Integer vector. Years for scenario period (default 2085:2095).
calib_years	Integer vector. Years for calibration period (default 1991:2020).
scen_warming	Numeric. Warming in degC (simple shift model).
shift_per_deg	Numeric. DOY shift per degC (simple shift).
cmip6_tas	Optional list of daily temperature vectors–only required if using `"gdd_cmip6"`.
gdd_base	Numeric. Base temperature for GDD model.
gdd_threshold1	Numeric. GDD threshold for phase1.
gdd_threshold2	Numeric. GDD threshold for phase2.
regions	Character vector. Countries/regions to include. If NULL, all are used.
delta_Tgl	Numeric. Global temperature change (degC) for scenario.
giss_data	Data frame with GISS temperature data (columns: `year`, `dT` or `Tgl`). Required for robust_shift method.

Value

A data frame containing:

Country	Country name
doy_start_ctrl	Start DOY for CTRL period
doy_end_ctrl	End DOY for CTRL period
doy_start_scen	Start DOY for SCEN period
doy_end_scen	End DOY for SCEN period

Author(s)

Matthias Templ

Examples

```
pep <- pep_download()

# Use Alpine subset for faster computation
regions <- c("Austria", "Switzerland")
pep_alpine <- pep[country %in% regions]

# Simple shift method
d1 <- get_phenology_doys(
  pep = pep_alpine,
  phase1 = 65,
  phase2 = 87,
  genus_name = "Malus",
  regions = regions,
  phenology_method = 'simple_shift'
)
d1
```

`is.pep` *Test if Object is a PEP725 Data Object*

Description

Test if Object is a PEP725 Data Object

Usage

```
is.pep(x)
```

Arguments

`x`	An R object.

Value

Logical; `TRUE` if `x` is of class pep.

Author(s)

Matthias Templ

Examples

```
pep <- pep_download()
is.pep(pep)
is.pep(data.frame(x = 1))
```

`kendall_tau` *Mann-Kendall Z-Statistic*

Description

Computes the Mann-Kendall Z-statistic for a time series, providing a non-parametric test of monotonic trend. The Z-statistic is approximately standard normal under the no-trend null hypothesis.

Usage

```
kendall_tau(x)
```

Arguments

`x`	Numeric vector (time series assumed to be equidistant). `NA` values are silently removed.

Value

Numeric. The Mann-Kendall Z-statistic (`S / sqrt(Var(S))`). Positive values indicate increasing trend, negative values decreasing. Compare against standard normal quantiles (e.g., $|Z| > 1.96$ for $p < 0.05$). Returns `NA` if fewer than 3 non-missing values.

Author(s)

Matthias Templ

Examples

```
# Decreasing trend (earlier phenology)
kendall_tau(c(120, 118, 115, 112, 110, 108, 105))

# No clear trend
kendall_tau(c(120, 115, 122, 118, 121, 116, 119))
```

new_pep *Create a PEP725 Phenological Data Object*

Description

Constructor function that creates a validated pep object from a data.frame or data.table. The object inherits from `data.table` and retains all its functionality.

Usage

```
new_pep(x, validate = TRUE)
```

Arguments

`x`	A data.frame or data.table containing PEP725 phenological data.
`validate`	Logical. If `TRUE` (default), validates that required columns are present and have correct types.

Details

Required columns:

- `s_id` - Station ID
- `lon`, `lat` - Coordinates
- `genus`, `species` - Plant taxonomy
- `phase_id` - BBCH phenological phase code
- `year`, `day` - Observation timing

Value

An object of class `c("pep", "data.table", "data.frame")`.

Author(s)

Matthias Templ

Examples

```
# From imported data
dt <- pep_download()
pep_data <- new_pep(dt)
```

`pep-class` *PEP725 Phenological Data Class*

Description

S3 class for PEP725 phenological data that inherits from `data.table`. Provides validation, informative print/summary methods, and convenient defaults.

`pep725_demo` *Demonstrate the pep725 Package Functions*

Description

This function provides an interactive demonstration of the main features of the pep725 package for PEP725 phenological data analysis. It showcases data exploration, filtering, summarization, and visualization capabilities.

Usage

```
pep725_demo(which = "all", pause = interactive())
```

Arguments

`which`	Character vector specifying which demos to run. Options are: • `"class"` - Demonstrate the pep class (print, summary) • `"filter"` - Show data filtering and selection • `"plot"` - Display visualization capabilities • `"analysis"` - Show analysis functions (normals, anomalies) • `"all"` - Run all demonstrations (default)
`pause`	Logical. If `TRUE` (default), pauses between demos waiting for user input. Set to `FALSE` for non-interactive use.

Details

The demonstration downloads synthetic PEP725 data using `pep_download()` and walks through typical workflows for phenological data analysis:

1. **Class demo**: Shows the enhanced print and summary methods for the pep class, demonstrating how to quickly understand the data.
2. **Filter demo**: Demonstrates selecting specific species, phases, and time periods using data.table syntax.
3. **Plot demo**: Creates visualizations including station maps, time series, and DOY distributions.
4. **Analysis demo**: Shows phenological normals calculation, anomaly detection, and data quality assessment.

Value

Invisibly returns a list containing example outputs from each demo.

Author(s)

Matthias Templ

See Also

`pep_download` for downloading synthetic data, `pheno_normals` for calculating normals, `pheno_anomaly` for anomaly detection, `pep_quality` for quality assessment

Examples

```
if (interactive()) {
# Run all demos interactively
pep725_demo()

# Run specific demos
pep725_demo(which = "class")
pep725_demo(which = c("filter", "plot"))

# Run without pausing (for scripts)
pep725_demo(pause = FALSE)
}
```

`pep_cache_clear` *Clear Cached PEP Data*

Description

Removes cached synthetic PEP data from the local cache directory.

Usage

```
pep_cache_clear(quiet = FALSE)
```

Arguments

`quiet`	Logical. If `TRUE`, suppress messages. Default is `FALSE`.

Value

Invisible `TRUE` if cache was cleared, `FALSE` otherwise.

Author(s)

Matthias Templ

See Also

pep_download

Examples

```
# Clear the cache
pep_cache_clear()

# Next download will fetch fresh data
pep <- pep_download()
```

---

pep_cache_info *Get Cache Information*

Description

Returns information about the cached PEP data.

Usage

```
pep_cache_info()
```

Value

A list with cache information, or NULL if no cache exists.

Author(s)

Matthias Templ

Examples

```
# Check cache status
pep_cache_info()
```

`pep_check_connectivity`
Check Station-Year Connectivity

Description

Checks whether station-year combinations in phenological data form a connected set, which is required for valid combined time series estimation.

Usage

```
pep_check_connectivity(pep, by = NULL)
```

Arguments

`pep`	A `pep` object or data.table with `year`, `day`, and `s_id` columns.
`by`	Optional grouping variables to check connectivity within groups.

Details

Connectivity means that there exist overlapping years between stations, allowing year effects to be separated from station effects. If data is disconnected, the combined series cannot be uniquely estimated across the disconnected parts.

Value

A list with connectivity information:

is_connected Logical. TRUE if all data forms one connected set.

n_sets Number of disconnected sets found.

set_sizes Number of observations in each set.

largest_set Stations and years in the largest connected set.

Author(s)

Matthias Templ

See Also

`pheno_combine` which uses this internally

Examples

```
pep <- pep_download()
pep_alpine <- pep[country %in% c("Switzerland", "Austria")]
conn <- pep_check_connectivity(pep_alpine)
if (!conn$is_connected) {
  warning("Data has ", conn$n_sets, " disconnected sets")
}
```

---

pep_check_phases *Check for Expected Phenological Phases*

Description

Validates that expected phenological phases (BBCH codes) are present in the data. Issues informative warnings for missing phases without stopping execution, making it useful for pipeline validation.

Usage

```
pep_check_phases(
  pep,
  expected = NULL,
  species = NULL,
  year_min = NULL,
  year_max = NULL,
  warn = TRUE,
  label = "Data"
)
```

Arguments

`pep`	A pep object or `data.table` containing phenological observations with a `phase_id` column.
`expected`	Integer vector of expected BBCH phase codes. If `NULL` (default), checks for common phases: 10 (leaf), 60 (heading/flowering), 65 (full flowering), and 100 (harvest).
`species`	Optional character string to filter by species/genus before checking. If specified, only checks phases for this species.
`year_min`	Optional integer. Only check phases for years >= this value. Useful for checking data availability within an analysis window.
`year_max`	Optional integer. Only check phases for years <= this value.
`warn`	Logical. If `TRUE` (default), issues warnings for missing phases. If `FALSE`, only returns the result silently.
`label`	Character. Optional label to include in warning messages, useful when checking multiple datasets. Default is `"Data"`.

Details

This function is designed to be used at the start of analysis pipelines to validate that required phenological phases are available. It issues warnings but does not stop execution, allowing users to be informed of potential issues while still proceeding with available data.

Value

A list with class `phase_check` containing:

expected The expected phase codes

present Phase codes found in the data

missing Phase codes not found in the data

extra Phase codes in data but not in expected list

complete Logical. TRUE if all expected phases are present

n_obs Named integer vector of observation counts per present phase

coverage Data.table with phase-level statistics

Common BBCH Phases

10 Leaf development

60 Beginning of flowering / Heading

65 Full flowering / Anthesis

69 End of flowering

87 Hard dough (cereals)

100 Harvest

Author(s)

Matthias Templ

See Also

`bbch_description` for phase code descriptions, `pep_completeness` for detailed completeness assessment, `pep_quality` for quality grading

Examples

```
pep <- pep_download()

# Use Swiss subset for faster checking
pep_ch <- pep[country == "Switzerland"]

# Basic check for common phases
check <- pep_check_phases(pep_ch)

# Check specific phases for grapevine (longest historical records)
check <- pep_check_phases(pep_ch,
                      expected = c(60, 65, 81),
                      species = "Vitis vinifera")

# Check within analysis window
check <- pep_check_phases(pep_ch,
                      expected = c(60, 65),
                      year_min = 1991,
                      year_max = 2020)

# Use in pipeline with custom label
vine_data <- pep_ch[species == "Vitis vinifera"]
if (nrow(vine_data) > 0) {
  pep_check_phases(vine_data, expected = c(60, 65),
               label = "Grapevine analysis")
}

# Silent check (no warnings)
result <- pep_check_phases(pep_ch, warn = FALSE)
if (!result$complete) {
  # Handle missing phases programmatically
}
```

---

pep_check_phases_multi

Check Multiple Phases Across Species

Description

A convenience function to check phase availability across multiple species at once.

Usage

```
pep_check_phases_multi(
  pep,
  species_list,
  expected = c(60, 65, 100),
  year_min = NULL,
  year_max = NULL,
  warn = TRUE
)
```

Arguments

`pep`	A pep object or `data.table`
`species_list`	Character vector of species/genus names to check
`expected`	Integer vector of expected BBCH phase codes

`year_min`	Optional minimum year filter
`year_max`	Optional maximum year filter
`warn`	Logical. Issue warnings for missing phases? Default TRUE.

Value

A `data.table` summarizing phase availability per species

Author(s)

Matthias Templ

See Also

`pep_check_phases` for single-species checking

Examples

```
# Check multiple species (use subset for speed)
pep <- pep_download()
pep_subset <- pep[country %in% c("Switzerland", "Austria")]
result <- pep_check_phases_multi(pep_subset,
                             species_list = c("Malus domestica",
                                              "Vitis vinifera"),
                             expected = c(60, 65))
print(result)
```

`pep_completeness` *Assess Species and Phase Completeness of Phenological Data*

Description

Provides comprehensive assessment of data completeness across species, phases, years, and optionally countries. Useful for identifying data gaps and planning analyses.

Usage

```
pep_completeness(
  pep,
  by = c("genus", "species", "phase_id"),
  year_range = NULL,
  min_obs = 1,
  include_years = FALSE
)
```

Arguments

`pep`	A pep object or `data.table` containing phenological observations with columns `year`, `day`, `genus`, `species`, and `phase_id`.
`by`	Character vector specifying grouping dimensions. Options include `"species"`, `"genus"`, `"phase_id"`, `"year"`, `"country"`, and `"s_id"`. Default is `c("genus", "species", "phase_id")`.
`year_range`	Optional integer vector of length 2 specifying the start and end years for assessment. If `NULL` (default), uses the full range of years in the data.
`min_obs`	Integer. Minimum number of observations required to include a group in the output. Default is 1.
`include_years`	Logical. If `TRUE`, includes year-level detail in the output (creates a row per group × year). Default is `FALSE` for summary mode.

Details

This function assesses data coverage across multiple dimensions, helping users understand where data is available and where gaps exist. It is particularly useful for:

- Identifying which species × phase combinations have sufficient data
- Finding temporal gaps in time series
- Comparing coverage across countries or regions
- Planning analyses that require minimum observation counts

Value

A `data.table` of class `pep_completeness` with columns:

by variables Grouping variables as specified in `by`

n_obs Total number of observations

n_stations Number of unique stations

n_years Number of years with data

year_min First year of observations

year_max Last year of observations

year_span Total span in years

completeness_pct Percentage of years with data within span

median_doy Median day of year

iqr_doy Interquartile range of day of year

Completeness Calculation

Completeness is calculated as the percentage of years with at least one observation within the year span (year_max - year_min + 1). A completeness of 100% means every year in the span has data.

Author(s)

Matthias Templ

See Also

pep_check_phases for validating expected phases exist, pep_quality for quality grading, pep_coverage for overall data coverage

Examples

```
pep <- pep_download()

# Use Swiss subset for faster computation
pep_ch <- pep[country == "Switzerland"]

# Basic completeness by species and phase
comp <- pep_completeness(pep_ch)
print(comp)

# Filter to well-observed combinations
comp_good <- comp[n_obs >= 100 & completeness_pct >= 80]

# Completeness by country (use Alpine countries)
pep_alpine <- pep[country %in% c("Switzerland", "Austria")]
comp_country <- pep_completeness(pep_alpine,
                                 by = c("country", "genus", "phase_id"))

# Year-level detail for a specific period
comp_yearly <- pep_completeness(pep_ch,
                                 year_range = c(1991, 2020),
                                 include_years = TRUE)

# Visualize completeness
plot(comp)
```

---

pep_coverage *Assess Data Coverage of PEP725 Phenological Data*

Description

Provides comprehensive coverage statistics for phenological datasets, including temporal, geographical, and species dimensions.

Usage

```
pep_coverage(
  x,
  kind = c("all", "temporal", "geographical", "species"),
  top = 10,
  plot = FALSE,
  by = NULL
)
```

Arguments

`x`	A pep object or data.frame with phenological data.
`kind`	Character. Type of coverage to assess: • `"all"` (default): All coverage types combined • `"temporal"`: Year range, gaps, observations per year • `"geographical"`: Countries, stations, coordinate ranges • `"species"`: Species and genus diversity
`top`	Integer. Number of top entries to show in tables (default: 10).
`plot`	Logical. If `TRUE`, produces a visualization of the coverage. Default is `FALSE`.
`by`	Character. Optional grouping variable for more detailed analysis (e.g., `"country"`, `"species"`). Only used for temporal coverage.

Value

A list of class `pep_coverage` containing coverage statistics. The structure depends on `kind`:

temporal Year range, number of years, gaps, observations per year

geographical Countries, stations, coordinate bounds, altitude range

species Genera, species, subspecies counts and details

all All of the above combined

Author(s)

Matthias Templ

Examples

```
pep <- pep_download()
pep_ch <- pep[country == "Switzerland"]

# Full coverage report
pep_coverage(pep_ch)

# Temporal coverage only
pep_coverage(pep_ch, kind = "temporal")

# Species coverage, top 5
```

```
pep_coverage(pep_ch, kind = "species", top = 5)
```

`pep_download` *Download Synthetic PEP725 Data*

Description

Downloads a pre-generated synthetic version of PEP725 phenological data from an external repository. The data is cached locally after the first download to avoid repeated downloads.

Usage

```
pep_download(url = NULL, cache = TRUE, force = FALSE, quiet = FALSE)
```

Arguments

`url`	Character string specifying the URL to download from. Default uses the official pep725 package data repository.
`cache`	Logical. If `TRUE` (default), cache the downloaded data locally for future use.
`force`	Logical. If `TRUE`, force re-download even if cached data exists. Default is `FALSE`.
`quiet`	Logical. If `TRUE`, suppress download progress messages. Default is `FALSE`.

Details

The synthetic data preserves the statistical properties and structure of real PEP725 data (stations, species, phases, temporal patterns) while ensuring data privacy. It is suitable for:

- Learning and testing package functions
- Reproducing examples from vignettes
- Developing analysis workflows before accessing real data

The data is cached in the platform-appropriate user cache directory determined by `R_user_dir("pep725", which = "cache")`. Typical locations:

- Windows: `%LOCALAPPDATA%/R/cache/R/pep725/`
- macOS: `~/Library/Caches/org.R-project.R/R/pep725/`
- Linux: `~/.cache/R/pep725/`

Value

A pep object containing synthetic phenological data with the same structure as original PEP725 data.

Data Access

If you have access to real PEP725 data, you can import it using `pep_import` instead.

Author(s)

Matthias Templ

See Also

`pep_simulate` for creating your own synthetic data, `pep_import` for importing real PEP725 data

Examples

```
# Download synthetic data
pep <- pep_download()

# Explore the data
print(pep)
summary(pep)

# Force re-download
pep <- pep_download(force = TRUE)
```

`pep_flag_outliers` *Flag Phenological Outliers*

Description

Identifies outlier observations in phenological data using the 30-day rule or statistical methods. Based on the approach described by Schaber & Badeck (2002) for quality control of phenological observations.

Usage

```
pep_flag_outliers(
  pep,
  by = c("s_id", "genus", "species", "phase_id"),
  method = c("30day", "mad", "iqr", "zscore"),
  threshold = NULL,
  center = c("median", "mean"),
  min_obs = 5,
  flag_only = TRUE
)
```

Arguments

`pep`	A pep object or data.table with phenological observations. Must contain `year`, `day`, and grouping columns.

`by`	Character vector of columns defining groups for outlier detection. Default `c("s_id", "genus", "species", "phase_id")` detects outliers within each station-species-phase combination.
`method`	Character. Outlier detection method: **"30day"** (Default) Flag observations >30 days from group mean/median. The classic Schaber-Badeck approach for phenological data. **"mad"** Use Median Absolute Deviation. Flag if |deviation| > k * MAD, where k is set by `threshold`. **"iqr"** Use Interquartile Range. Flag if outside (Q1 - k*IQR, Q3 + k*IQR). **"zscore"** Flag if |z-score| > threshold (default 3).
`threshold`	Numeric. Threshold for outlier detection. **For "30day":** Days deviation to flag as outlier. Default 30. **For "mad":** Number of MADs. Default 3 (~2.5 SD equivalent). **For "iqr":** IQR multiplier. Default 1.5 (standard Tukey rule). **For "zscore":** Z-score threshold. Default 3.
`center`	Character. Central tendency measure: `"median"` (default, more robust) or `"mean"`.
`min_obs`	Integer. Minimum observations per group required for outlier detection. Groups with fewer observations are skipped. Default 5.
`flag_only`	Logical. If `TRUE` (default), adds outlier flag columns to data. If `FALSE`, removes flagged outliers from data.

Details

The 30-day rule (Schaber & Badeck 2002) is widely used for phenological data quality control. It flags observations that deviate more than 30 days from the expected value for that station-species-phase combination. This threshold is based on the observation that legitimate phenological variation rarely exceeds one month.

Value

If `flag_only = TRUE`: The input data with additional columns:

is_outlier Logical. TRUE for flagged outliers.

deviation Numeric. Days deviation from expected value.

expected_doy Numeric. Expected DOY for the group.

If `flag_only = FALSE`: Data with outliers removed.

When to Use Each Method

30day Standard for phenological QC. Use when the 30-day biological threshold is meaningful.

mad Robust to existing outliers. Good for initial screening of potentially contaminated data.

iqr Standard statistical approach. Useful when comparing with other quality metrics.

zscore Parametric approach. Use when data is approximately normal.

Author(s)

Matthias Templ

References

Schaber J, Badeck F-W (2002). Evaluation of methods for the combination of phenological time series and outlier detection. Tree Physiology 22:973-982.

See Also

`pep_quality` for comprehensive quality assessment, `pheno_combine` which uses residuals for outlier detection

Examples

```
data(pep_seed)

# Flag outliers using 30-day rule
pep_flagged <- pep_flag_outliers(pep_seed)
table(pep_flagged$is_outlier)

# View flagged observations
pep_flagged[is_outlier == TRUE]

# Remove outliers instead of flagging
pep_clean <- pep_flag_outliers(pep_seed, flag_only = FALSE)

# Use MAD method with stricter threshold
pep_flagged <- pep_flag_outliers(pep_seed, method = "mad", threshold = 2.5)

# Group by country instead of station
pep_flagged <- pep_flag_outliers(pep_seed,
                              by = c("country", "genus", "phase_id"))
```

---

pep_import   *Import and preprocess PEP725 phenological data*

Description

This function imports all CSV files from the specified folder, reads them efficiently using **data.table**, and combines them into a single data table. It performs several preprocessing steps, including:

- Converting selected columns to factors (`provider_id`, `s_id`, `gss_id`, `genus`, `species`, `subspecies`).
- Replacing missing altitude values coded as `-9999` with `NA`.
- Creating a combined altitude variable `alt2` using `alt` and `alt_dem`.
- Converting the `date` column to `Date` class.

- Recoding cultivation season (`cult_season`) and quality control flags (`qc_ori_flag`) into labeled factors.
- Removing unused or problematic columns (`affected_flag`, `qc_flag`).

Usage

```
pep_import(path = "data/Data_PEP725_all", flags = FALSE, add_country = TRUE)
```

Arguments

`path`	path to the folder containing PEP725 CSV files (default is `"data/Data_PEP725_all/"`).
`flags`	Logical indicating whether the pep data contains quality control flags (default is `FALSE`).
`add_country`	Logical indicating whether to add country information based on station coordinates (default is `TRUE`).

Details

This function is intended for importing raw PEP725 station data into a standardized format suitable for further phenological analysis, such as sensitivity and trend estimation.

Value

A `pep` object (extends `data.table`) containing the combined and preprocessed PEP725 data.

Author(s)

Matthias Templ (FHNW)

See Also

`fread`, `rbindlist`

Examples

```
# Small example files shipped with the package
path <- system.file("extdata", package = "pep725")
pep_data <- pep_import(path = path, add_country = FALSE)
print(pep_data)
```

pep_plot_outliers *Visualize Phenological Outliers for Inspection*

Description

Creates diagnostic plots to help distinguish between data errors and biologically meaningful extreme events (e.g., second flowering). Multiple plot types reveal different aspects of outlier patterns.

Usage

```
pep_plot_outliers(
  x,
  type = c("overview", "seasonal", "map", "detail", "station", "doy_context"),
  phase_id = NULL,
  outlier_only = NULL,
  late_threshold = 250,
  n_top = 20,
  ...
)
```

Arguments

`x`	A `pep_outliers` object from `pep_flag_outliers`.
`type`	Character. Type of plot to produce: **"overview"** (Default) Multi-panel overview with distribution, seasonal pattern, and summary statistics. **"seasonal"** Distribution of outliers by month/season, useful for detecting second flowering (late-season outliers for spring phases). **"map"** Geographic distribution of outliers (requires lon/lat). **"detail"** Detailed view of individual outlier events with context. **"station"** Station-level outlier patterns over time. **"doy_context"** Shows outliers in context of full DOY distribution per phase, highlighting potential second events.
`phase_id`	Optional integer vector to filter specific phases for plotting.
`outlier_only`	Logical. If TRUE (default for some types), show only outlier observations. If FALSE, show all data with outliers highlighted.
`late_threshold`	Integer. DOY threshold for classifying "late" outliers as potential second events (default 250, ~September 7).
`n_top`	Integer. For detail view, number of most extreme outliers to show. Default 20.
`...`	Additional arguments passed to plotting functions.

Details

This function is designed to support quality control workflows where the goal is to distinguish:

1. **Data errors**: Typos, wrong dates, mis-coded phases
2. **Observer errors**: Misidentified species or phases
3. **Biologically meaningful extremes**: Second flowering, delayed development, climate-driven anomalies

For detecting potential **second flowering events**:

- Use `type = "seasonal"` to see if late-season outliers cluster in autumn months
- Use `type = "doy_context"` to visualize outliers relative to the main flowering distribution
- Focus on flowering phases (60, 65) with late DOY values (>250)

Value

A `ggplot` object (or list of ggplots for "overview").

Author(s)

Matthias Templ

See Also

`pep_flag_outliers` for detecting outliers, `pep_quality` for comprehensive quality assessment

Examples

```
pep <- pep_download()

# Use Swiss subset for faster outlier detection
pep_ch <- pep[country == "Switzerland"]
pep_flagged <- pep_flag_outliers(pep_ch)

# Overview of all outliers
pep_plot_outliers(pep_flagged, type = "overview")

# Seasonal distribution - look for autumn outliers in flowering phases
pep_plot_outliers(pep_flagged, type = "seasonal", phase_id = c(60, 65))

# Geographic pattern
pep_plot_outliers(pep_flagged, type = "map")

# Detailed view of most extreme outliers
pep_plot_outliers(pep_flagged, type = "detail", n_top = 30)

# DOY context for flowering phase - spot potential second flowering
pep_plot_outliers(pep_flagged, type = "doy_context", phase_id = 65)
```

pep_quality — *Assess Data Quality of Phenological Observations*

Description

Provides a comprehensive assessment of data completeness, temporal coverage, outlier prevalence, and assigns quality grades to phenological time series.

Usage

```
pep_quality(
  pep,
  by = c("s_id", "genus", "phase_id"),
  year_range = NULL,
  species = NULL,
  phase_id = NULL,
  outlier_method = c("tukey", "zscore"),
  na.rm = TRUE
)
```

Arguments

`pep`	A pep object or `data.table` containing phenological observations with columns `year`, `day`, and grouping variables.
`by`	Character vector of column names to group by for quality assessment. Default is `c("s_id", "genus", "phase_id")` for station-level assessment.
`year_range`	Optional integer vector of length 2 specifying the start and end years for assessment. If `NULL` (default), uses the full range of years in the data.
`species`	Optional character string to filter by species/genus.
`phase_id`	Optional integer vector to filter by BBCH phase codes.
`outlier_method`	Character string specifying outlier detection method. Either `"tukey"` (default, uses 1.5*IQR rule) or `"zscore"` (flags values with $\|z\| > 3$).
`na.rm`	Logical. Should missing values be removed? Default `TRUE`.

Details

This function evaluates phenological time series quality based on multiple criteria important for trend analysis and climatological calculations.

Value

A `data.table` with the following columns:

by variables Grouping variables as specified in by

n_obs Total number of observations

year_min First year of observations

year_max Last year of observations

year_span Total span in years (year_max - year_min + 1)

n_years Number of distinct years with data

completeness_pct Percentage of years with data within the span

max_gap_years Length of the longest gap (consecutive missing years)

n_gaps Number of gaps in the time series

n_outliers Number of statistical outliers detected

outlier_pct Percentage of observations that are outliers

doy_mean Mean day-of-year

doy_sd Standard deviation of day-of-year

doy_range Range of day-of-year values

quality_grade Overall quality grade (A, B, C, or D)

Quality Grading Criteria

Grade A Completeness >= 80%, no gaps > 2 years, outliers < 2%

Grade B Completeness >= 60%, no gaps > 5 years, outliers < 5%

Grade C Completeness >= 40%, gaps <= 10 years, outliers < 10%

Grade D Below grade C thresholds

Outlier Detection

The Tukey method (default) flags observations outside the interquartile fences: `[Q1 - 1.5*IQR, Q3 + 1.5*IQR]`. This is robust to non-normal distributions common in phenological data.

Author(s)

Matthias Templ

References

Schaber, J. and Badeck, F.-W. (2002). Evaluation of methods for the combination of phenological time series and outlier detection. *Tree Physiology*, 22, 973–982. doi:10.1093/treephys/22.14.973

See Also

`pheno_normals` for calculating climatological normals, `pheno_anomaly` for anomaly detection

Examples

```
pep <- pep_download()

# Subset to one country for speed
pep_ch <- pep[country == "Switzerland"]

# Assess quality for stations in Switzerland
quality <- pep_quality(pep_ch)
print(quality)

# Filter to high-quality stations only
high_quality <- quality[quality_grade %in% c("A", "B")]
high_quality

# Assess specific species and phase
apple_quality <- pep_quality(pep_ch,
                             species = "Malus",
                             phase_id = 60)
apple_quality

# Country-level assessment (use subset of countries)
pep_subset <- pep[country %in% c("Switzerland", "Austria")]
country_quality <- pep_quality(pep_subset,
                               by = c("country", "genus", "phase_id"))
country_quality

# Assess quality within a specific time window
modern_quality <- pep_quality(pep_ch,
                              year_range = c(1991, 2020))
modern_quality
```

`pep_second_events` *Detect Second Flowering and Other Repeated Phenological Events*

Description

Identifies potential second flowering events and other irregular repeated phenological observations. These may represent biologically meaningful responses to climate disruption rather than data errors.

Usage

```
pep_second_events(
  pep,
  phases = c(60, 65),
  method = c("late_season", "multiple_per_year", "both"),
  late_threshold = 250,
  gap_threshold = 60,
  reference_period = 1991:2020,
  z_threshold = 3,
  min_obs = 10
)
```

Arguments

`pep`	A pep object or data.table with phenological observations.
`phases`	Integer vector of phase IDs to analyze. Default is `c(60, 65)` for flowering phases. Use `NULL` to analyze all phases.
`method`	Character. Detection method: **"late_season"** (Default) Flag observations occurring after the expected phenological window (DOY > late_threshold). **"multiple_per_year"** Detect multiple observations of the same phase at the same station within a single year. **"both"** Apply both methods.
`late_threshold`	Integer. DOY threshold for "late season" events. Default is 250 (~September 7). Events after this date for spring phases are flagged as potential second events.
`gap_threshold`	Integer. For "multiple_per_year" method, minimum days between observations to consider them separate events. Default is 60.
`reference_period`	Integer vector of years to calculate expected phenological windows. Default is `1991:2020`.
`z_threshold`	Numeric. Z-score threshold for identifying late events relative to the reference period. Default is 3.
`min_obs`	Integer. Minimum observations in reference period to calculate expected windows. Default is 10.

Details

Second flowering and other repeated phenological events are increasingly reported as climate patterns become more irregular. These events may indicate:

- Disrupted coordination between environmental cues and development
- Response to unusual autumn warmth after summer dormancy
- Failure of normal seasonal inhibition mechanisms

The function uses two complementary approaches:

1. **Late season detection**: Identifies observations occurring well after the normal phenological window (e.g., flowering in autumn)
2. **Multiple events**: Finds stations reporting the same phase multiple times in one year with sufficient gap between observations

Value

A list of class `second_events` containing:

events Data.table of detected second/repeated events with columns: s_id, species, phase_id, year, day, event_type, expected timing info

summary Summary statistics by species, phase, and event type

by_year Annual counts of detected events

by_month Monthly distribution of events

total_by_year Total observations per year (for relative scaling in plots)

method Detection method used

params Parameters used for detection

Interpreting Results

Not all detected events are necessarily "true" second flowering. Consider:

- **High confidence**: Same station reports phase twice with 60+ day gap; late observation is clearly outside normal window
- **Medium confidence**: Single late observation at a station; could be second event or data entry error
- **Requires verification**: Check original data sources, geographic context, species biology

Author(s)

Matthias Templ

References

Related to emerging research on phenological irregularity and climate disruption of seasonal timing.

See Also

`pep_flag_outliers` for general outlier detection, `pep_plot_outliers` for outlier visualization

Examples

```
pep <- pep_download()

# Use grapevine flowering in Alpine region (long records, second events possible)
pep_subset <- pep[species == "Vitis vinifera" & phase_id %in% c(60, 65) &
                  country %in% c("Germany-South", "Switzerland", "Austria")]

if (nrow(pep_subset) > 0) {
  # Method 1: Detect late-season flowering events (default)
  second <- pep_second_events(pep_subset, phases = c(60, 65))
  print(second)
  summary(second)
  # Plot from 1980 onwards for clearer trends
  plot(second, from_year = 1980)
  plot(second, type = "overview", scale = "relative", from_year = 1980)

  # Method 2: Detect repeated observations at same station
  repeated <- pep_second_events(pep_subset, method = "multiple_per_year")
  print(repeated)
  plot(repeated, type = "timeline", from_year = 1980)

  # Method 3: Combine both detection methods
  all_events <- pep_second_events(pep_subset, method = "both")
  summary(all_events)
  plot(all_events)
  plot(all_events, scale = "relative")
}
```

pep_seed *Minimal Seed Dataset for Synthetic Data Generation*

Description

A small subset of phenological data suitable for use with pep_simulate to generate synthetic phenological datasets. This dataset contains real structure but limited scope, making it suitable for package examples and testing.

Format

A data.table with 1,319 rows and 10 variables:

s_id Station identifier (factor)

lon Longitude in decimal degrees

lat Latitude in decimal degrees

alt Altitude in meters

genus Plant genus (factor)

species Full species name (factor)

phase_id BBCH phenological phase code (integer)

year Observation year

day Day of year (DOY) of the phenological event

country Country name

Details

This dataset includes observations for:

- **Species**: Triticum aestivum (wheat), Vitis vinifera (grapevine), Malus domestica (apple)
- **Phases**: BBCH 10 (emergence), 60 (flowering), 65 (full flowering)
- **Countries**: Austria, Germany-South
- **Years**: 1990-2015
- **Stations**: 30 stations

For a full synthetic dataset, use `pep_download` to download pre-generated synthetic data from the package repository.

Source

Derived from PEP725 Pan European Phenology Database `http://www.pep725.eu/`

See Also

`pep_simulate` for generating synthetic data, `pep_download` for downloading full synthetic dataset

Examples

```
# Load the seed dataset
data(pep_seed)

# Examine structure
str(pep_seed)

# Use with pep_simulate to create synthetic data

pep_synthetic <- pep_simulate(pep_seed)
```

`pep_simulate` *Simulate Synthetic PEP Data*

Description

Generates synthetic phenological observations by fitting a GAM smooth (`day ~ s(year)`) per station-species-phase group and replacing the observed day values with predictions plus Gaussian noise. The station-year-species-phase structure of the input data is kept intact.

Usage

```
pep_simulate(pep, min_obs = 20, seed = 42, progress = TRUE)
```

Arguments

`pep`	A `pep` object or `data.table` with at least the columns `species`, `phase_id`, `s_id`, `year`, and `day`.
`min_obs`	Integer. Minimum number of observations required per station-species-phase group to attempt GAM fitting. Groups with fewer observations keep their original `day` values unchanged. Default is 20.
`seed`	Integer. Random seed for reproducibility. Default is 42.
`progress`	Logical. Show a progress bar? Default `TRUE`.

Details

For each qualifying group (station x species x phase with at least `min_obs` observations), a GAM is fitted and synthetic day-of-year values are generated as `round(predicted + rnorm(n, sd = residual_sd))`.

Groups where the GAM fit fails receive a fallback: the species-phase median day plus Gaussian jitter (sd = 5 days).

Groups below the `min_obs` threshold are left unchanged. A message reports how many rows were not simulated.

Value

A `pep` object with the same rows and columns as the input. The `day` column is overwritten with synthetic values for groups that meet the `min_obs` threshold; other rows retain their original values.

Author(s)

Matthias Templ

Examples

```
pep <- pep_download()
# Single species + country for fast example
vine_ch <- pep[species == "Vitis vinifera" & country == "Switzerland"]
pep_synth <- pep_simulate(vine_ch)
```

`pheno_anomaly` *Calculate Phenological Anomalies*

Description

Computes deviations from long-term phenological baselines to identify extreme years and detect climate signals. Returns both raw anomalies (in days) and standardized anomalies for cross-species comparison.

Usage

```
pheno_anomaly(
  pep,
  baseline_period = 1961:1990,
  target_years = NULL,
  by = c("country", "genus", "phase_id"),
  species = NULL,
  phase_id = NULL,
  robust = TRUE,
  min_years = 15,
  extreme_threshold = 2,
  normals = NULL,
  na.rm = TRUE
)
```

Arguments

`pep`	A pep object or `data.table` containing phenological observations with columns `year`, `day`, and grouping variables.
`baseline_period`	Integer vector specifying the years to use for baseline calculation. Default is `1961:1990` (pre-acceleration warming reference period).
`target_years`	Integer vector specifying years to calculate anomalies for. If `NULL` (default), calculates for all years in the data.
`by`	Character vector of column names to group by. Must match columns in the data. Default is `c("country", "genus", "phase_id")`.
`species`	Optional character string to filter by species/genus. If `NULL` (default), all species are included.
`phase_id`	Optional integer vector to filter by BBCH phase codes. If `NULL` (default), all phases are included.
`robust`	Logical. If `TRUE` (default), uses median and MAD for baseline and standardization. If `FALSE`, uses mean and SD.
`min_years`	Minimum number of years required in the baseline period to calculate valid anomalies. Default is 15.
`extreme_threshold`	Numeric. Z-score threshold for flagging extreme events. Default is 2 (approximately 95th percentile for normal distribution).
`normals`	Optional pre-computed `pheno_normals` object. If provided, baseline statistics are taken from this object instead of being computed.
`na.rm`	Logical. Should missing values be removed? Default `TRUE`.

Details

Phenological anomalies quantify how much a given year deviates from long-term expectations. This is crucial for:

- Detecting climate change signals in phenology
- Identifying extreme phenological years
- Comparing anomalies across different species/regions

The function supports two approaches:

- **Robust** (default): Uses median for central tendency and MAD (Median Absolute Deviation) for spread. More resistant to outliers.
- **Classical**: Uses mean and standard deviation. More sensitive to extreme values but provides interpretable z-scores under normality.

Value

A `data.table` with the following columns:

by variables Grouping variables as specified in `by`

year Year of observation

observed_doy Observed mean day-of-year for the group/year

baseline_doy Long-term baseline (median if robust, mean otherwise)

baseline_spread Baseline spread (MAD if robust, SD otherwise)

anomaly_days Deviation in days (negative = early, positive = late)

z_score Standardized anomaly (anomaly / spread)

percentile Percentile rank relative to baseline distribution

is_extreme Logical flag for extreme events (|z_score| > threshold)

direction Character: "early", "late", or "normal"

Interpretation

- **anomaly_days**: Direct interpretation - "X days earlier/later than normal"
- **z_score**: Standardized - values > 2 or < -2 are typically considered extreme
- **percentile**: Normal approximation from z-score - e.g., 5th percentile means "earlier than 95\

Author(s)

Matthias Templ

References

Menzel, A. et al. (2006). European phenological response to climate change matches the warming pattern. *Global Change Biology*, 12(10), 1969–1976. doi:10.1111/j.13652486.2006.01193.x

Templ, M., Templ, B., and Filzmoser, P. (2018). Modelling and prediction of phenological data of the beginning of flowering: Austrian indicators of climate change. *International Journal of Biometeorology*, 62, 1319–1334. doi:10.1007/s0048401815342

See Also

`pheno_normals` for calculating baseline statistics, `pep_download` for obtaining the main dataset

Examples

```
pep <- pep_download()

# Grapevine in Switzerland (longest historical records back to 1775)
vine_ch <- pep[country == "Switzerland" & species == "Vitis vinifera"]

if (nrow(vine_ch) > 0) {
  # Calculate anomalies relative to 1961-1990 baseline
  anomalies <- pheno_anomaly(vine_ch,
                              baseline_period = 1961:1990,
                              phase_id = 65)
  print(anomalies)

  # Find extreme early years
  extreme_early <- anomalies[is_extreme == TRUE & direction == "early"]
  extreme_early

  # Anomalies for recent years only
  recent <- pheno_anomaly(vine_ch,
                           baseline_period = 1961:1990,
                           target_years = 2000:2020,
                           phase_id = 65)
  recent
}
```

---

pheno_combine *Create Combined Phenological Time Series*

Description

Estimates a combined (regional/national) phenological time series from multi-station observations by separating year effects from station effects. This is essential for creating representative time series from networks with varying station coverage over time.

Usage

```
pheno_combine(
  pep,
  by = NULL,
  method = c("robust", "mixed", "ols"),
  min_years = 5,
  min_stations = 3,
  check_connectivity = TRUE
)
```

Arguments

`pep`	A pep object or `data.table` containing phenological observations with columns `year`, `day`, and `s_id`.
`by`	Character vector of grouping variables (e.g., `c("genus", "species", "phase_id")`). A separate combined series is created for each group. Default is `NULL` (treat all data as one group).
`method`	Character. Estimation method: **"robust"** (Default) Least Absolute Deviations (LAD/L1) regression. Minimizes sum of absolute residuals, robust to outliers. **"mixed"** Mixed-effects model with year as fixed effect and station as random effect. Uses REML estimation. **"ols"** Ordinary Least Squares. Sensitive to outliers but provides standard errors and confidence intervals.
`min_years`	Integer. Minimum years of overlap required. Default 5.
`min_stations`	Integer. Minimum stations required. Default 3.
`check_connectivity`	Logical. If `TRUE` (default), checks whether station-year combinations form a connected set. Warns if data is disconnected (which prevents valid combined series estimation).

Details

The model fitted is:

$$DOY_{ij} = \mu_i + \sigma_j + \epsilon_{ij}$$

where μ_i is the year effect (combined series), σ_j is the station effect, and ϵ_{ij} is the residual.

Station effects represent systematic differences between stations (e.g., due to altitude, microclimate, or observer differences). The combined series removes these effects to reveal the underlying temporal trend.

Value

An object of class `pheno_combined` containing:

- **combined** Data.table with the combined time series: year, year_effect (the combined DOY), se (standard error, if available)
- **station_effects** Data.table of station effects: s_id, effect, se
- **residuals** Data.table with original data plus fitted values and residuals for outlier detection
- **method** Estimation method used
- **connectivity** Connectivity check results (if performed)
- **by** Grouping variables used
- **group_results** For grouped analysis, results per group

Connectivity

A prerequisite for valid estimation is that station-year combinations form a "connected" set - meaning there must be overlapping years between stations to separate year and station effects. The function checks this automatically and warns if data is disconnected.

Outlier Detection

With `method = "robust"`, large residuals (e.g., > 30 days) indicate potential outliers or data errors. The residuals table can be used to identify and investigate these.

Author(s)

Matthias Templ

References

Schaber J, Badeck F-W (2002). Evaluation of methods for the combination of phenological time series and outlier detection. Tree Physiology 22:973-982.

See Also

`pheno_normals` for climatological baselines, `pheno_trend_turning` for trend turning point detection, `pep_check_connectivity` for connectivity assessment

Examples

```
pep <- pep_download()

# Grapevine flowering in Switzerland (longest historical records)
vine_ch <- pep[species == "Vitis vinifera" &
               phase_id == 65 & country == "Switzerland"]

if (nrow(vine_ch) > 0) {
  # Create combined series using OLS
  combined <- pheno_combine(vine_ch, method = "ols")
  print(combined)

  # Identify potential outliers (residuals > 30 days)
  outliers <- combined$residuals[abs(residual) > 30]
  outliers
}
```

pheno_gradient — *Analyze Phenological Gradients with Elevation or Latitude*

Description

Quantifies how phenological timing shifts with altitude or latitude, calculating the phenological lapse rate using robust regression methods.

Usage

```
pheno_gradient(
  pep,
  variable = c("alt", "lat"),
  species = NULL,
  phase_id = NULL,
  year_range = NULL,
  by = NULL,
  method = c("robust", "ols", "quantile"),
  aggregate = TRUE,
  na.rm = TRUE
)
```

Arguments

`pep`	A pep object or `data.table` containing phenological observations with columns `year`, `day`, and either `alt` (altitude) or `lat` (latitude).
`variable`	Character string specifying the gradient variable. Either `"alt"` (altitude/elevation) or `"lat"` (latitude). Default is `"alt"`.
`species`	Optional character string to filter by species/genus.
`phase_id`	Optional integer to filter by a single BBCH phase code.
`year_range`	Optional integer vector of length 2 specifying years to include.
`by`	Optional character vector of grouping variables (e.g., "country"). If provided, separate gradients are calculated for each group.
`method`	Character string specifying regression method: • `"robust"` (default): Robust regression using `robustbase::lmrob` • `"ols"`: Ordinary least squares regression • `"quantile"`: Quantile regression for median using `quantreg::rq`
`aggregate`	Logical. If `TRUE` (default), first aggregate to station means before fitting. If `FALSE`, use all individual observations.
`na.rm`	Logical. Should missing values be removed? Default `TRUE`.

Details

Phenological gradients describe how the timing of biological events changes with environmental factors. This function calculates:

- **Altitude gradient**: Typically 2-4 days delay per 100m elevation gain in temperate regions
- **Latitude gradient**: Typically 2-5 days delay per degree north in Europe

The robust regression method (default) is recommended because phenological data often contains outliers from observation errors or microclimate effects.

Value

A list with class `pheno_gradient` containing:

summary A `data.table` with slope, intercept, R-squared, RMSE, n_stations, and interpretation

model The fitted model object(s)

data The data used for fitting

variable The gradient variable used

method The regression method used

Interpretation

The slope represents:

- For altitude: days delay per 100 meters elevation increase
- For latitude: days delay per degree latitude increase

Positive slopes indicate later phenology at higher elevations/latitudes.

Author(s)

Matthias Templ

References

Defila, C. and Clot, B. (2001). Phytophenological trends in Switzerland. *International Journal of Biometeorology*, 45, 203–207. doi:10.1007/s004840100101

Ziello, C. et al. (2009). Influence of altitude on phenology of selected plant species in the Alpine region (1971–2000). *Climate Research*, 39, 227–234. doi:10.3354/cr00822

See Also

`pheno_normals` for calculating climatological normals, `lmrob` for robust regression details

Examples

```
pep <- pep_download()

# Subset to Alpine countries (good elevation variation)
pep_subset <- pep[country %in% c("Switzerland", "Austria")]

# Altitude gradient for apple flowering
grad_alt <- pheno_gradient(pep_subset,
                           variable = "alt",
                           species = "Malus",
                           phase_id = 60)
print(grad_alt)

# Latitude gradient
grad_lat <- pheno_gradient(pep_subset,
                           variable = "lat",
                           species = "Malus",
                           phase_id = 60)
print(grad_lat)

# Compare regression methods
grad_ols <- pheno_gradient(pep_subset, species = "Malus",
                           phase_id = 60, method = "ols")
print(grad_ols)

grad_robust <- pheno_gradient(pep_subset, species = "Malus",
                              phase_id = 60, method = "robust")
print(grad_robust)

# Gradient by country (separate regression per country)
grad_by_country <- pheno_gradient(pep_subset,
                                  variable = "alt",
                                  species = "Malus",
                                  phase_id = 60,
                                  by = "country")
print(grad_by_country)
```

`pheno_leaflet` *Interactive Leaflet Map to Select PEP725 Stations*

Description

Launches a Shiny gadget that allows selection of PEP phenological stations by drawing a bounding box or clicking individual points on the map.

Usage

```
pheno_leaflet(pep, label_col = NULL, quiet = FALSE)
```

Arguments

`pep`	A data frame or data.table with columns `lon`, `lat`, and optionally `species`.
`label_col`	Optional column name to show as label in popups (e.g. `"species"` or `"s_id"`).
`quiet`	Logical. If `FALSE` (default), displays a message about expected loading time.

Details

Note: Loading time depends on the size of the dataset. For the full PEP725 dataset with millions of observations, expect 10-20 seconds for the map to fully render. Consider filtering the data before calling this function to speed up loading:

```
# Filter to a specific species first
wheat <- pep[genus == "Triticum"]
selected <- pheno_leaflet(wheat)
```

Value

A `data.frame` with selected stations (columns: `lon`, `lat`, and metadata).

Author(s)

Matthias Templ

Examples

```
if (interactive()) {
pep <- pep_download()

# For faster loading, filter the data first (recommended)
# Grapevine has the longest historical records
pep_subset <- pep[species == "Vitis vinifera"]

if (nrow(pep_subset) > 0) {
  # Launch interactive selection gadget
  selected <- pheno_leaflet(pep_subset, label_col = "species")

  # Inspect selected subset
  print(selected)
}
}
```

pheno_map *Plot Phenology Station Maps*

Description

Generates a map of PEP725 phenological station locations with optional coloring by various statistics including mean phenological timing, trends, and species variation.

Usage

```
pheno_map(
  pep,
  location = c(lon = 6.233, lat = 46.4),
  zoom = 4,
  background = c("google", "none"),
  color_by = c("none", "n_obs", "n_species", "mean_doy", "trend", "species_cv"),
  phase_id = NULL,
  period = NULL,
  min_years = 10,
  min_species = 3,
  point_size = 0.8,
  output_file = NULL,
  key = NULL
)
```

Arguments

`pep` A `data.table` or `data.frame` with phenological observations. Required columns depend on `color_by`:

- Always required: `lon`, `lat`
- For `"n_species"`, `"mean_doy"`, `"species_cv"`: `species`
- For `"mean_doy"`, `"trend"`: `year`, `day`

`location` Named vector with center longitude and latitude for the map. Defaults to near Changins: `c(lon = 6.233, lat = 46.400)`. Used primarily for `background = "google"`.

`zoom` Zoom level for the map. For `background = "google"`, this is the Google Maps zoom level (4 = Europe, 7 = regional). For `background = "none"`, this controls the padding around data extent (4 = wide view, 7 = tight view). Default is 4.

`background` Character. Map background type:

"google" Google Maps satellite/terrain background. Requires API key (set via `ggmap::register_google()` or `key` parameter).

"none" Simple map with country borders from Natural Earth data. No API key required. Good for publications and vignettes.

`color_by` Character. What to color stations by:

	"none" Black points showing station locations only **"n_obs"** Number of observations per station **"n_species"** Number of species recorded per station **"mean_doy"** Mean day-of-year (phenological timing) per station **"trend"** Trend in days/year per station (positive = later, negative = earlier) **"species_cv"** Coefficient of variation in timing across species at each station (higher = more variation among species)
`phase_id`	Integer. BBCH phase code to filter by. Recommended for `"mean_doy"` and `"trend"` to ensure meaningful comparisons. If `NULL` (default), all phases are included.
`period`	Integer vector of years to include. Default is all years. For trend calculation, at least 10 years are recommended.
`min_years`	Integer. Minimum years required for trend calculation. Stations with fewer years show as NA. Default is 10.
`min_species`	Integer. Minimum species required for `species_cv`. Default is 3.
`point_size`	Size of station points (default: 0.8).
`output_file`	Optional file path to export the plot (e.g. `"map.pdf"`).
`key`	Google Maps API key. Only needed for `background = "google"`. You can set globally via `ggmap::register_google()` instead.

Details

The function supports two background types:

Google Maps background (`background = "google"`): Provides detailed satellite or terrain imagery but requires a Google Maps API key. Register at `https://console.cloud.google.com/` and enable the Maps Static API.

No background (`background = "none"`): Uses country outlines from Natural Earth data. No API key required, making it suitable for:

- CRAN package vignettes (no external API calls)
- Reproducible research (no authentication needed)
- Publication-quality maps with clean appearance

The color options provide insights into:

- **mean_doy**: Spatial patterns in phenological timing. Earlier timing (lower DOY) typically in southern/lowland areas.
- **trend**: Where phenology is advancing (negative = earlier over time, shown in blue) or delaying (positive, shown in red). Uses Kendall's tau normalized statistic for robustness.
- **species_cv**: Stations where different species show similar timing (low CV) vs. divergent timing (high CV). High variation may indicate species-specific responses to local conditions.

Value

A ggplot map object.

Interpreting Trends

The trend is calculated as Kendall's normalized tau statistic, which ranges roughly from -3 to +3 for typical data:

- Negative values (blue): Phenology is getting earlier over time
- Positive values (red): Phenology is getting later over time
- Values near 0: No clear trend
- |tau| > 1.96: Statistically significant at 95% level

Author(s)

Matthias Templ

See Also

`pheno_normals` for detailed normal calculations, `kendall_tau` for trend calculation, `pheno_synchrony` for synchrony analysis

Examples

```
pep <- pep_download()
pep_alpine <- pep[country %in% c("Switzerland", "Austria")]
pheno_map(pep_alpine, background = "none", color_by = "n_obs")

## Not run:
# With Google Maps background (requires API key)
ggmap::register_google(key = "your_api_key_here")
pheno_map(pep, background = "google", color_by = "n_species", zoom = 5)

## End(Not run)
```

`pheno_normals` *Calculate Phenological Normals (Climatology)*

Description

Computes reference "normal" phenology for a specified period, analogous to WMO climate normals. Returns central tendency, spread, and percentile statistics for phenological day-of-year (DOY) values.

Usage

```
pheno_normals(
  pep,
  period = 1991:2020,
  by = c("country", "genus", "phase_id"),
  species = NULL,
  phase_id = NULL,
  min_years = 20,
  probs = c(0.05, 0.1, 0.25, 0.75, 0.9, 0.95),
  na.rm = TRUE
)
```

Arguments

`pep`	A pep object or `data.table` containing phenological observations with columns `year`, `day`, and grouping variables.
`period`	Integer vector specifying the years to include in the normal calculation. Default is `1991:2020` (current WMO standard period).
`by`	Character vector of column names to group by. Common choices: • `c("country", "genus", "phase_id")` - Regional normals by genus/phase • `c("s_id", "species", "phase_id")` - Station-level normals • `c("genus", "phase_id")` - Overall normals ignoring geography. Default is `c("country", "genus", "phase_id")`.
`species`	Optional character string to filter by species column. Can be genus name (e.g., "Triticum") or full species (e.g., "Triticum aestivum"). If `NULL` (default), all species in the data are included.
`phase_id`	Optional integer vector to filter by BBCH phase codes. If `NULL` (default), all phases in the data are included.
`min_years`	Minimum number of years required to calculate valid normals. Default is 20 (WMO standard). Groups with fewer years return `NA`.
`probs`	Numeric vector of probabilities for percentile calculation. Default is `c(0.05, 0.10, 0.25, 0.75, 0.90, 0.95)`.
`na.rm`	Logical. Should missing values be removed? Default `TRUE`.

Details

Phenological normals provide a baseline for comparing individual years or detecting trends. This function calculates both classical statistics (mean, SD) and robust alternatives (median, MAD, IQR) that are less sensitive to outliers.

The function can be used in two ways:

1. **Full dataset**: Pass the complete `pep` object and use `species` and `phase_id` parameters to filter internally.
2. **Pre-filtered subset**: Filter the data first using data.table syntax, then pass the subset to the function.

Value

A `data.table` with the following columns:

by variables Grouping variables as specified in `by`

n_years Number of years with data in the period

n_obs Total number of observations

mean_doy Arithmetic mean DOY

median_doy Median DOY (more robust to outliers)

sd_doy Standard deviation

iqr_doy Interquartile range

mad_doy Median absolute deviation (robust spread)

q05, q10, q25, q75, q90, q95 Percentiles (or as specified in `probs`)

period Character string describing the reference period

Standard Reference Periods

- **1961-1990**: Historical reference (pre-acceleration of warming)
- **1991-2020**: Current WMO standard normal period

Author(s)

Matthias Templ

References

WMO (2017). *WMO Guidelines on the Calculation of Climate Normals*. WMO-No. 1203, World Meteorological Organization, Geneva.

Templ, M., Templ, B., and Filzmoser, P. (2018). Modelling and prediction of phenological data of the beginning of flowering: Austrian indicators of climate change. *International Journal of Biometeorology*, 62, 1319–1334. doi:10.1007/s0048401815342

See Also

`pheno_anomaly` for calculating anomalies relative to normals, `pep_download` for obtaining the main dataset

Examples

```
# Download synthetic data first
pep <- pep_download()

# Use Swiss subset for faster computation
pep_ch <- pep[country == "Switzerland"]

# Calculate normals for all species and phases
normals_all <- pheno_normals(pep_ch)

# Normals for apples, flowering phase (60)
apple_normals <- pheno_normals(pep_ch,
                               species = "Malus",
                               phase_id = 60)

# Using a pre-filtered subset
```

```
apples <- pep_ch[genus == "Malus" & phase_id %in% c(60, 65)]
apple_normals <- pheno_normals(apples,
                               by = c("species", "phase_id"))

# Station-level normals for detailed analysis
station_normals <- pheno_normals(pep_ch,
                                 species = "Malus",
                                 phase_id = 60,
                                 by = c("s_id"))

# Historical reference period
historical <- pheno_normals(pep_ch,
                            period = 1961:1990,
                            species = "Malus")

# Compare two periods
period1 <- pheno_normals(pep_ch, period = 1961:1990, species = "Malus")
period2 <- pheno_normals(pep_ch, period = 1991:2020, species = "Malus")
shift <- period2$mean_doy - period1$mean_doy
```

pheno_plot *Plot Phenological Time Series*

Description

Generates time series plots from processed phenology data. Shows DOY trends with faceting by phenophase and spatial scope.

Usage

```
pheno_plot(data_list, alpha_lines = 0.6, linewidth = 0.7)
```

Arguments

`data_list`	A named list of prepared data objects, typically output from `pheno_regional`. Must contain element `ts_tidy`.
`alpha_lines`	Alpha transparency for time series lines (default is `0.6`).
`linewidth`	Line width for time series lines (default is `0.7`).

Details

This function visualizes DOY trends with faceting by phenophase and spatial scope.

Value

A `ggplot` object for visual inspection and further customization.

Author(s)

Matthias Templ

See Also

`pheno_regional`

Examples

```
# Construct a minimal data_list with ts_tidy
ts_tidy <- data.frame(
  year = rep(1990:1995, 2),
  DOY = c(120, 118, 115, 119, 113, 110, 125, 122, 120, 124, 118, 115),
  source = rep(c("PEP725 (aggregated)", "PEP725 (near Changins)"), each = 6),
  phase = "Heading",
  panel = "DOY"
)
pheno_plot(list(ts_tidy = ts_tidy))
```

`pheno_plot_hh` *Plot Phenological Time Series for Heading and Harvest*

Description

Visualizes phenological time series for heading and harvest phases, comparing aggregated and spatially filtered PEP725 data.

Usage

```
pheno_plot_hh(
  data_list,
  phase_select = NULL,
  alpha_lines = 0.6,
  linewidth = 0.7
)
```

Arguments

`data_list`	A named list returned by `pheno_regional_hh`, containing `ts_tidy`.
`phase_select`	Character. Optional filter for a specific phenological phase (e.g., "Heading", "Harvest"). If NULL, all phases are shown.
`alpha_lines`	Numeric. Transparency level for lines (default is 0.6).
`linewidth`	Numeric. Line width for time series plots (default is 0.7).

Details

This function shows DOY trends by phase and data source across years, with month labels on the y-axis for easier interpretation.

Value

A `ggplot` object.

Author(s)

Matthias Templ

See Also

`pheno_regional_hh`

Examples

```
pep <- pep_download()
out <- pheno_regional_hh(pep,
        species_name = "Triticum aestivum")
pheno_plot_hh(out)
```

`pheno_plot_timeseries` *Plot Phenological Time Series*

Description

Visualizes phenological day-of-year (DOY) trends over time since 1961. Designed for aggregated or site-level phenology data from PEP725.

Usage

```
pheno_plot_timeseries(
  data,
  color_by = "species",
  facet_by = NULL,
  alpha_lines = 0.7,
  linewidth = 0.8,
  smooth = TRUE,
  se = FALSE,
  year_min = 1961,
  title = "Phenological time series of flowering (BBCH 65)"
)
```

Arguments

`data`	A data.frame or data.table containing at least `year`, `day` (day-of-year), and one grouping variable (e.g. `species`, `phase`, or `functional_group`).
`color_by`	Character. Column name used for color grouping (default = "species").
`facet_by`	Character or NULL. Optional column name for faceting (e.g. "functional_group").
`alpha_lines`	Numeric. Transparency level for time-series lines (default = 0.7).
`linewidth`	Numeric. Line width (default = 0.8).
`smooth`	Logical. If TRUE, adds a linear trend smoother (default = TRUE).
`se`	Logical. Whether to display confidence interval for smoother (default = FALSE).
`year_min`	Numeric. Minimum year to plot (default = 1961).
`title`	Character. Optional plot title.

Value

A ggplot object.

Author(s)

Matthias Templ

Examples

```
pep <- pep_download()

# Use Alpine subset for faster computation
pep_alpine <- pep[country %in% c("Switzerland", "Austria")]

# Example: flowering DOY by genus
pheno_plot_timeseries(data = pep_alpine[phase_id == 65],
                      color_by = "genus",
                      facet_by = NULL, smooth = TRUE)

# Example: single species, colored by site (grapevine has longest records)
vine_data <- pep_alpine[species == "Vitis vinifera" & phase_id == 65]
if (nrow(vine_data) > 0) {
  pheno_plot_timeseries(
    data = vine_data,
    color_by = "s_id",
    smooth = TRUE,
    title = "Grapevine flowering (BBCH 65) across sites"
  )
}

# Example: facets by genus
pheno_plot_timeseries(
  data = pep_alpine[phase_id == 65],
  color_by = "species",
  facet_by = "genus",
  smooth = TRUE
)
```

pheno_pls — *Partial Least Squares Analysis for Phenology-Temperature Relationships*

Description

Performs PLS regression to identify temperature-sensitive periods affecting phenological timing. This is useful for understanding which parts of the year have the strongest influence on phenological events.

Usage

```
pheno_pls(
  pep,
  temp_data,
  by = NULL,
  method = c("robust", "standard"),
  split_month = 6,
  runn_mean = 11,
  ncomp = NULL,
  expl_var = 30,
  max_iter = 20,
  tol = 1e-04
)
```

Arguments

`pep`	A `pep` object or data.frame with phenological observations. Must contain `year` and `day` columns.
`temp_data`	A data.frame with daily temperature data. Must contain columns `year`, `doy` (day of year), and either `Tmean` or both `Tmin` and `Tmax`.
`by`	Character vector of grouping columns to match phenology and temperature data (e.g., `"s_id"` for station-specific analysis). Default `NULL` uses the same temperature for all observations per year.
`method`	Character. PLS method to use: **"robust"** (Default) Iteratively reweighted PLS using bisquare weights. Resistant to outliers in both phenology and temperature data. **"standard"** Classical PLS regression without robust weighting.
`split_month`	Integer (1-12). The last month to include in the phenological year. Default 6 (June) means July-June phenological years. This is important for spring phenology where relevant temperatures span the previous autumn/winter.

`runn_mean`	Integer (odd). Window size for running mean smoothing of temperatures. Default 11 days. Larger values produce smoother results.
`ncomp`	Integer or NULL. Number of PLS components. If NULL (default), automatically determined to explain at least `expl_var` percent of variance.
`expl_var`	Numeric. Minimum percentage of variance to explain when automatically selecting components. Default 30.
`max_iter`	Integer. Maximum iterations for robust method. Default 20.
`tol`	Numeric. Convergence tolerance for robust method. Default 1e-4.

Details

PLS regression is particularly useful for phenology-temperature analysis because it handles the high collinearity between daily temperatures (adjacent days are highly correlated) while identifying which periods have the strongest influence on phenological timing.

The robust method uses iteratively reweighted PLS with bisquare weights, which downweights observations with large residuals. This is important for phenological data which may contain outliers from observation errors or unusual weather events.

Value

An object of class `pheno_pls` containing:

vip Data.frame with Variable Importance in Projection scores for each day of the phenological year.

coefficients PLS regression coefficients for each day.

model The fitted PLS model object.

weights Observation weights (all 1 for standard method).

r_squared Model R-squared.

ncomp Number of components used.

method Method used ("robust" or "standard").

n_obs Number of observations.

n_years Number of years analyzed.

Interpreting Results

VIP scores Values > 0.8 indicate important variables. Days with high VIP scores have strong influence on phenology timing.

Coefficients Negative coefficients indicate that warmer temperatures during that period lead to earlier phenology (lower DOY).

Phenological Year

Spring phenology (e.g., flowering) is influenced by temperatures from the previous autumn through spring. The `split_month` parameter defines where to split the calendar year. With `split_month = 6`, a flowering event in April 2020 is analyzed against temperatures from July 2019 through June 2020.

Author(s)

Matthias Templ

References

Luedeling E, Gassner A (2012). Partial Least Squares Regression for analyzing walnut phenology in California. Agricultural and Forest Meteorology 158-159:104-113.

Wold S, Sjostrom M, Eriksson L (2001). PLS-regression: a basic tool of chemometrics. Chemometrics and Intelligent Laboratory Systems 58:109-130.

See Also

`pheno_trend_turning` for trend analysis, `pheno_anomaly` for anomaly detection

Examples

```
data(pep_seed)

# Create example temperature data (fewer years for speed)
years <- 2005:2012
temp <- data.frame(
  year = rep(years, each = 365),
  doy = rep(1:365, length(years)),
  Tmin = rnorm(365 * length(years), mean = 5, sd = 8),
  Tmax = rnorm(365 * length(years), mean = 15, sd = 8)
)

# Run standard PLS (fast: no CV, fixed components)
result <- pheno_pls(pep_seed, temp, method = "standard", ncomp = 2)
print(result)
```

`pheno_regional` *Compile Regional Phenology Data and Climate Sensitivity Inputs*

Description

Extracts and aggregates phenological observations for a selected species and phenological phase from the PEP725 dataset (globally and spatially filtered). Optionally links them with global temperature anomalies (GISS data) for climate impact studies.

Usage

```
pheno_regional(
  pep,
  giss = NULL,
  lon_min = 4.2,
  lon_max = 8.1,
  lat_min = 44.7,
  lat_max = 48.1,
  species_name = "Triticum aestivum",
  functional_group = NULL,
  year_min = 1961,
  pep_for_giss = c("near", "aggregated"),
  phase = 60
)
```

Arguments

`pep`	A data.table containing the full PEP725 dataset. Must include at least columns: `species`, `year`, `phase_id`, `day`, `lat`, `lon`.
`giss`	Optional. A data.table with GISS global temperature anomalies, containing columns `year`, `dT`, and `dT_sm`. If `NULL`, GISS-related outputs are omitted.
`lon_min`	Minimum longitude for spatial filtering of PEP data (default is `4.2`)
`lon_max`	Maximum longitude for spatial filtering of PEP data (default is `8.1`)
`lat_min`	Minimum latitude for spatial filtering of PEP data (default is `44.7`)
`lat_max`	Maximum latitude for spatial filtering of PEP data (default is `48.1`)
`species_name`	Character name of the species to be extracted (default is `"Triticum aestivum"`)
`functional_group`	Character. Optional functional group name to filter by (e.g., "C4_summer_cereals"). If provided, overwrites species_name filtering.
`year_min`	Minimum year to include in all outputs (default is `1961`)
`pep_for_giss`	Selects which PEP data subset to use for merging with GISS data. Either `"near"` (box-filtered) or `"aggregated"` (global). Ignored if `giss` is `NULL`.
`phase`	Integer or vector of phase_id(s) to extract (default is `60` = Heading). Named phase mappings are applied.

Details

The function aggregates PEP observations by species, year, and phase, calculates mean day-of-year (DOY), applies spatial filtering if needed, and attaches GISS climate anomalies. It also maps `phase_ids` to descriptive names and warns if expected phenological phase are missing in the selected data.

If GISS data is provided, temperature anomalies are merged with the selected PEP time series to support phenology-climate analysis.

Value

A named list with the following elements:

`pep_agg`	Global (non-spatially filtered) PEP725 time series for the selected phase(s)
`pep_cgi`	Spatially filtered PEP725 time series based on lat/lon bounding box
`giss`	Processed GISS global temperature anomaly data (if provided)

`pep_giss` Merged data frame of PEP phenology and GISS anomalies (if provided)

`species` Character species name used for filtering

`functional_group` Functional group if specified, otherwise `NA`

`phase` Integer phase ID(s) used for filtering

Author(s)

Matthias Templ

See Also

`pheno_plot`, `pep_download`

Examples

```
pep <- pep_download()

out <- pheno_regional(pep, species_name = "Triticum aestivum", phase = 60)
str(out)
```

`pheno_regional_hh` *Compile Regional Phenological Time Series for Heading and Harvest*

Description

Generates harmonized phenological time series for heading (BBCH 60) and harvest (BBCH 100) phases of a specified species and spatial region. Optionally integrates GISS climate data for climate-sensitivity studies.

Usage

```
pheno_regional_hh(
  pep,
  giss = NULL,
  lon_min = 4.2,
  lon_max = 8.1,
  lat_min = 44.7,
  lat_max = 48.1,
  species_name = "Triticum aestivum",
  year_min = 1961,
  pep_for_giss = c("aggregated", "near")
)
```

Arguments

`pep`	A data.table containing PEP725 phenological data.
`giss`	Optional. A data.table with GISS global temperature anomalies (`year`, `dT`, `dT_sm`). If `NULL`, GISS-related outputs are omitted.
`lon_min`	Minimum longitude for bounding box filter (default is 4.2)
`lon_max`	Maximum longitude for bounding box filter (default is 8.1)
`lat_min`	Minimum latitude for bounding box filter (default is 44.7)
`lat_max`	Maximum latitude for bounding box filter (default is 48.1)
`species_name`	Name of the species to filter (default is `"Triticum aestivum"`)
`year_min`	Minimum year for all time series outputs (default is 1961)
`pep_for_giss`	Which PEP725 dataset to use for merging with GISS climate anomalies. Either `"aggregated"` (global) or `"near"` (spatially filtered). Ignored if `giss` is `NULL`.

Details

This function filters the PEP725 dataset by species and optionally by spatial box. It aggregates phenological phases (Heading = 60, Harvest = 100) and computes mean day-of-year (DOY) per year. If GISS data is provided, temperature anomalies are merged with the selected PEP time series to support phenology-climate analysis. Warnings are issued if phases are missing for selected periods.

Value

A named list with the following elements:

`ts_tidy`	Long-format time series with consistent phase and source labeling
`pep_agg`	Aggregated PEP725 time series (global species-level)
`pep_cgi`	Spatially filtered PEP725 time series (based on lat/lon box)
`giss`	GISS global temperature anomalies (if provided)
`pep_giss`	PEP time series merged with GISS anomalies (if provided)
`species`	Species name used for filtering

Author(s)

Matthias Templ

See Also

`pep_import`, `pheno_plot_hh`

Examples

```
pep <- pep_download()

out <- pheno_regional_hh(pep,
         species_name = "Triticum aestivum")
pheno_plot_hh(out)
```

pheno_synchrony *Analyze Phenological Synchrony Across Stations*

Description

Measures how synchronized phenological events are across stations within regions and whether synchrony is changing over time. Higher synchrony indicates more uniform timing of phenological events.

Usage

```
pheno_synchrony(
  pep,
  species = NULL,
  phase_id = NULL,
  by = c("country", "year"),
  min_stations = 5,
  compute_trend = TRUE,
  na.rm = TRUE
)
```

Arguments

`pep`	A pep object or `data.table` containing phenological observations with columns `year`, `day`, and `s_id` (station ID).
`species`	Optional character string to filter by species/genus.
`phase_id`	Optional integer to filter by a single BBCH phase code.
`by`	Character vector of column names to group by for synchrony calculation. Default is `c("country", "year")` for annual synchrony by country.
`min_stations`	Minimum number of stations required to calculate synchrony. Default is 5. Groups with fewer stations return NA.
`compute_trend`	Logical. If `TRUE` (default), computes trend in synchrony over time using robust regression.
`na.rm`	Logical. Should missing values be removed? Default `TRUE`.

Details

Synchrony measures how similar phenological timing is across different stations within the same region and year. This is important for:

- Understanding spatial coherence of phenological signals
- Detecting changes in spatial variability over time
- Assessing network representativeness

Value

A `pheno_synchrony` object (list) containing:

data A `data.table` with synchrony metrics per group: n_stations, mean_doy, sd_doy, cv_pct, and quality indicators

trend If `compute_trend = TRUE`, a data.table with trend analysis results per region (slope, p-value, direction)

overall Summary statistics across all groups

Synchrony Metrics

sd_doy Standard deviation across stations - lower values indicate higher synchrony

cv_pct Coefficient of variation (SD/mean * 100) - relative measure that allows comparison across phases with different mean timing

range_doy Range of DOY values across stations

Trend Interpretation

- Negative trend in SD: Increasing synchrony over time
- Positive trend in SD: Decreasing synchrony (more variable)

Author(s)

Matthias Templ

References

Rosenzweig, C. et al. (2008). Attributing physical and biological impacts to anthropogenic climate change. *Nature*, 453, 353–357. doi:10.1038/nature06937

See Also

`pheno_normals` for climatological statistics, `pheno_anomaly` for anomaly detection

Examples

```
pep <- pep_download()

# Subset to two countries for speed
pep_subset <- pep[country %in% c("Switzerland", "Austria")]

# Calculate synchrony for apple flowering by country and year
sync <- pheno_synchrony(pep_subset,
                        species = "Malus",
                        phase_id = 60)
print(sync)

# Get trend results (robust regression per country)
sync$trend
```

```
# Synchrony without trend analysis (faster)
sync_simple <- pheno_synchrony(pep_subset,
                               species = "Malus",
                               phase_id = 60,
                               compute_trend = FALSE)
sync_simple

# Custom grouping variables
sync_detailed <- pheno_synchrony(pep_subset,
                                 species = "Malus",
                                 phase_id = 60,
                                 by = c("country", "year"))
sync_detailed
```

`pheno_trend_turning` *Detect Trend Turning Points in Phenological Time Series*

Description

Applies a sequential Mann-Kendall test to detect approximate trend turning points in phenological time series. This identifies years where trends may have changed direction (e.g., when spring advancement accelerated or reversed).

Usage

```
pheno_trend_turning(pep, by = NULL, min_years = 10, aggregate = TRUE)
```

Arguments

`pep`	A pep object, data.table, or numeric vector of DOY values. If a data.frame/data.table, must contain `year` and `day` columns.
`by`	Character vector of column names to group by before analysis. Default is `NULL` (analyze all data as one series). Common choices: `c("genus", "species", "phase_id")` or `c("country", "phase_id")`.
`min_years`	Integer. Minimum number of years required for analysis. Default is 10. Shorter series may produce unreliable results.
`aggregate`	Logical. If `TRUE` (default), aggregates multiple observations per year (e.g., from multiple stations) using the median. If `FALSE`, expects exactly one value per year.

Details

The sequential Mann-Kendall test calculates two series:

- **Progressive**: Kendall's tau computed from the start forward

- **Retrograde**: Kendall's tau computed from the end backward

Points where these series cross indicate potential trend turning points. When either series exceeds confidence thresholds (|tau| > 1.96 for 95%, |tau| > 2.58 for 99%) before and after a crossing, the turning point is considered statistically significant.

Value

An object of class `pheno_turning` containing:

results Data.table with columns: year, day (or median_day), tau_prog (progressive tau), tau_retr (retrograde tau), is_turning (logical indicating turning points)

turning_points Years identified as potential turning points

n_years Number of years in the series

by Grouping variables used (if any)

group_results If by is specified, list of results per group

Interpretation

- Positive tau indicates an increasing trend (later DOY = delayed phenology)
- Negative tau indicates a decreasing trend (earlier DOY = advanced phenology)
- Crossing points suggest the trend direction may have changed

Author(s)

Matthias Templ

References

Sneyers R (1990). On statistical analysis of series of observations. Technical Note No 143. World Meteorological Organization.

See Also

`pheno_gradient` for spatial trend analysis, `pheno_normals` for baseline calculations

Examples

```
# Simple vector input (fast)
doy_series <- c(120, 118, 122, 115, 110, 108, 112, 105, 102, 100)
turning <- pheno_trend_turning(doy_series)
print(turning)

# Using pep_seed data (no grouping for speed)
data(pep_seed)
vine <- pep_seed[pep_seed$species == "Vitis vinifera" &
                 pep_seed$phase_id == 65, ]
if (nrow(vine) > 0) {
  turning <- pheno_trend_turning(vine)
}
```

`plot.pep` *Plot Method for PEP725 Data*

Description

Provides default visualizations for PEP725 phenological data.

Usage

```
## S3 method for class 'pep'
plot(x, type = c("map", "timeseries", "histogram"), ...)
```

Arguments

`x`	A pep object.
`type`	Character. Type of plot: `"map"` for station locations, `"timeseries"` for temporal trends, `"histogram"` for DOY distribution. Default is `"map"`.
`...`	Additional arguments passed to the underlying plot function.

Value

A ggplot object (invisibly).

Author(s)

Matthias Templ

Examples

```
pep <- pep_download()
plot(pep)
plot(pep, type = "timeseries")
plot(pep, type = "histogram")
```

`plot.pep_completeness` *Plot Method for Completeness Assessment*

Description

Creates visualizations of data completeness, including heatmaps and bar charts.

Usage

```
## S3 method for class 'pep_completeness'
plot(x, type = c("heatmap", "bar", "timeline"), top = 20, ...)
```

Arguments

`x`	A `pep_completeness` object
`type`	Character. Type of plot: `"heatmap"` for completeness heatmap (requires 2 grouping variables), `"bar"` for observation counts, `"timeline"` for year coverage. Default is `"heatmap"`.
`top`	Integer. Number of top groups to display. Default is 20.
`...`	Additional arguments (unused)

Value

A ggplot object (invisibly)

Author(s)

Matthias Templ

Examples

```
pep <- pep_download()
pep_ch <- pep[country == "Switzerland"]
comp <- pep_completeness(pep_ch, by = c("genus", "phase_id"))
plot(comp, type = "heatmap")
plot(comp, type = "bar", top = 15)
```

`plot.pep_coverage` *Plot Method for PEP Coverage*

Description

Plot Method for PEP Coverage

Usage

```
## S3 method for class 'pep_coverage'
plot(x, ...)
```

Arguments

`x`	A `pep_coverage` object.
`...`	Additional arguments (unused).

Value

A ggplot object (or list of ggplot objects).

Author(s)

Matthias Templ

`plot.pep_outliers` *Plot Method for Outlier Detection Results*

Description

Plot Method for Outlier Detection Results

Usage

```
## S3 method for class 'pep_outliers'
plot(x, type = c("histogram", "scatter", "timeline"), ...)
```

Arguments

`x`	A `pep_outliers` object
`type`	Character. Plot type: "histogram" for deviation distribution, "scatter" for DOY vs expected, "timeline" for outliers over time.
`...`	Additional arguments

Value

A ggplot object (invisibly)

`plot.pep_quality` *Plot Method for Data Quality Assessment*

Description

Creates visualizations of phenological data quality metrics, including grade distribution and a map of station quality.

Usage

```
## S3 method for class 'pep_quality'
plot(
  x,
  which = c("overview", "grades", "map"),
  pep = NULL,
  show_grades = c("A", "B", "C", "D"),
  alpha = 0.6,
  title = NULL,
  ...
)
```

Arguments

`x`	A `pep_quality` object.
`which`	Character. Type of plot to produce: • `"overview"` (default): Two-panel overview with grade distribution and station quality map (requires pep argument) • `"grades"`: Bar chart of quality grade distribution only • `"map"`: Map of station locations colored by quality grade (requires pep argument for coordinates)
`pep`	Optional pep object providing station coordinates (lon, lat). Required for `which = "map"` or `which = "overview"`.
`show_grades`	Character vector of grades to display on the map. Default is `c("A", "B", "C", "D")` (all grades). Use e.g. `show_grades = "D"` to show only poor-quality stations.
`alpha`	Numeric. Transparency of points on the map (0-1). Default is 0.6.
`title`	Optional character string for the plot title.
`...`	Additional arguments (unused).

Details

For the map visualization, the function:

- Aggregates quality to one grade per station (worst grade if multiple phases)
- Uses country borders from the `rnaturalearth` package
- Uses colorblind-friendly colors (blue=A, cyan=B, orange=C, vermillion=D)

Value

A ggplot object (invisibly).

Author(s)

Matthias Templ

Examples

```
pep <- pep_download()

# Assess quality for Swiss stations
pep_ch <- pep[country == "Switzerland"]
quality <- pep_quality(pep_ch, by = c("s_id", "phase_id"))

# Grade distribution only (no pep data needed)
plot(quality, which = "grades")

# Map of station quality (requires pep for coordinates)
plot(quality, which = "map", pep = pep_ch)
```

```
# Map showing only poor-quality stations
plot(quality, which = "map", pep = pep_ch, show_grades = "D")

# Map showing problematic stations (C and D grades)
plot(quality, which = "map", pep = pep_ch, show_grades = c("C", "D"))

# Overview: grades + map
plot(quality, pep = pep_ch)
```

`plot.pheno_anomaly` *Plot Method for Phenological Anomalies*

Description

Visualise phenological anomalies as a colour-coded timeline or as a histogram of anomaly magnitudes.

Usage

```
## S3 method for class 'pheno_anomaly'
plot(x, which = c("timeline", "histogram"), ...)
```

Arguments

`x`	A `pheno_anomaly` object returned by `pheno_anomaly`.
`which`	Character string selecting the plot type: `"timeline"` (default) shows `anomaly_days` per year as bars coloured by `direction` with extreme events marked; `"histogram"` shows the distribution of anomaly values.
`...`	Additional arguments (unused).

Value

A `ggplot` object (returned invisibly).

Author(s)

Matthias Templ

See Also

`pheno_anomaly`, `pheno_normals`

Examples

```
pep <- pep_download()
vine_ch <- pep[country == "Switzerland" & species == "Vitis vinifera"]

if (nrow(vine_ch) > 0) {
  a <- pheno_anomaly(vine_ch,
                     baseline_period = 1961:1990,
                     phase_id = 65,
                     by = "phase_id")
  plot(a)
  plot(a, which = "histogram")
}
```

---

plot.pheno_combined *Plot Method for Combined Time Series*

Description

Plot Method for Combined Time Series

Usage

```
## S3 method for class 'pheno_combined'
plot(x, type = c("series", "stations", "residuals"), group = NULL, ...)
```

Arguments

x	A pheno_combined object
type	Character. "series" for combined time series, "stations" for station effects, "residuals" for residual distribution.
group	For grouped analysis, which group to plot.
...	Additional arguments

Value

A ggplot object (invisibly)

plot.pheno_gradient — *Plot Method for Phenological Gradient Analysis*

Description

Plot Method for Phenological Gradient Analysis

Usage

```
## S3 method for class 'pheno_gradient'
plot(x, ...)
```

Arguments

`x`	A `pheno_gradient` object
`...`	Additional arguments passed to ggplot

Value

A ggplot object

Author(s)

Matthias Templ

plot.pheno_normals — *Plot Method for Phenological Normals*

Description

Visualise phenological normals as a dot plot with interquartile-range bars or as a bar chart.

Usage

```
## S3 method for class 'pheno_normals'
plot(x, which = c("dotplot", "bar"), ...)
```

Arguments

`x`	A `pheno_normals` object returned by `pheno_normals`.
`which`	Character string selecting the plot type: `"dotplot"` (default) shows `median_doy` with Q25–Q75 range via `geom_pointrange`; `"bar"` shows `median_doy` as columns with error bars.
`...`	Additional arguments (unused).

Value

A ggplot object (returned invisibly).

Author(s)

Matthias Templ

See Also

pheno_normals, pheno_anomaly

Examples

```
pep <- pep_download()
pep_ch <- pep[country == "Switzerland"]

# Normals by phase for apple
n <- pheno_normals(pep_ch, species = "Malus", by = "phase_id")
plot(n)
plot(n, which = "bar")
```

---

plot.pheno_pls *Plot Method for PLS Phenology Results*

Description

Plot Method for PLS Phenology Results

Usage

```
## S3 method for class 'pheno_pls'
plot(x, type = c("vip", "coef", "both"), vip_threshold = 0.8, ...)
```

Arguments

x	A pheno_pls object
type	Character. Plot type: "vip" for VIP scores (default), "coef" for coefficients, "both" for combined plot.
vip_threshold	Numeric. Threshold line for VIP plot. Default 0.8.
...	Additional arguments passed to plotting functions

Value

A ggplot object (invisibly)

`plot.pheno_synchrony` *Plot Method for Phenological Synchrony Analysis*

Description

Creates a time series plot of synchrony (SD) over years.

Usage

```
## S3 method for class 'pheno_synchrony'
plot(x, region = NULL, ...)
```

Arguments

`x`	A `pheno_synchrony` object
`region`	Optional region to plot (for multi-region analyses)
`...`	Additional arguments passed to ggplot

Value

A ggplot object

Author(s)

Matthias Templ

`plot.pheno_turning` *Plot Method for Trend Turning Analysis*

Description

Plot Method for Trend Turning Analysis

Usage

```
## S3 method for class 'pheno_turning'
plot(x, group = NULL, show_thresholds = TRUE, ...)
```

Arguments

`x`	A `pheno_turning` object
`group`	Character. For grouped analysis, which group to plot. If `NULL` (default), plots the first group or ungrouped results.
`show_thresholds`	Logical. Show significance threshold lines? Default `TRUE`.
`...`	Additional arguments passed to ggplot

Value

A ggplot object (invisibly)

`plot.second_events` *Plot Method for Second Events Detection*

Description

Plot Method for Second Events Detection

Usage

```
## S3 method for class 'second_events'
plot(
  x,
  type = c("overview", "timeline", "seasonal", "map"),
  scale = c("absolute", "relative"),
  from_year = NULL,
  ...
)
```

Arguments

`x`	A `second_events` object
`type`	Character. Plot type: "overview", "timeline", "seasonal", "map"
`scale`	Character. Scale for y-axis in timeline/overview plots: **"absolute"** (Default) Show raw counts of second events per year. **"relative"** Show proportion of second events relative to total observations per year. This accounts for varying data availability over time and reveals whether second events are becoming proportionally more common.
`from_year`	Integer. If specified, only show data from this year onwards in timeline and overview plots. Useful for focusing on recent trends. Default is `NULL` (show all years).
`...`	Additional arguments

Value

A ggplot object (invisibly)

plot_phenology_trends *Plot Robust Phenological Trends*

Description

This function visualizes long-term phenological trends for one genus or species, using robust MM-regression (robustbase::lmrob) applied to mean annual DOYs.

Usage

```
plot_phenology_trends(
  pep,
  genus_name = NULL,
  species_name = NULL,
  subspecies_name = NULL,
  phases = c(65, 87),
  common_stations = TRUE,
  combine_regions = FALSE,
  combine_layout = c("vertical", "horizontal"),
  years = 1961:2024,
  calib_years = 1991:2020,
  pred_years = NULL,
  subregions = c("Austria", "Germany-North", "Germany-South", "Switzerland"),
  giss_data = NULL,
  layout = c("country_phase", "phase_country"),
  title = NULL
)
```

Arguments

pep	A PEP725-style data frame with columns year, DOY, phase_id, country, s_id, genus, species.
genus_name	Character string specifying a genus to filter (optional).
species_name	Character string specifying a species to filter (optional). If both genus and species are provided, the species filter is applied last.
subspecies_name	Character string specifying a subspecies to filter (optional). Only one of genus_name, species_name, or subspecies_name can be specified.
phases	Integer vector of phenological phase IDs to include (default: c(65, 87) for flowering and maturity of fruit).
common_stations	Logical. If TRUE (default), only stations with observations for all selected phases are included. If FALSE, all stations with any of the selected phases are included.
combine_regions	Logical. If TRUE, combines all regions into a single panel. Default is FALSE.

`combine_layout`	Character. Layout for combined regions plot: either `"vertical"` or `"horizontal"`.
`years`	Numeric vector of years to include in the analysis. Default is `1961:2024`.
`calib_years`	Numeric vector specifying the calibration window for robust regression (default: `1991:2020`).
`pred_years`	Numeric vector of years for projecting future phenology. Default is `1990:2090`.
`subregions`	Character vector of country names to include (default: DACH region).
`giss_data`	Deprecated. Previously used for climate data integration; now ignored.
`layout`	Either `"country_phase"` (default) or `"phase_country"` to control facet arrangement.
`title`	Optional plot title. If `NULL`, the function generates one from genus and species information.

Details

The function produces a faceted panel plot showing:

- annual mean DOY and interquartile range,
- robust trend lines for past and future periods,
- CTRL (1991-2020) and PGW (2066-2095) climate windows,
- optional layouts (country x phase or phase x country).

The function performs the following steps:

1. Filter PEP data by species, phases, regions, years.
2. Aggregate DOYs by year-country-phase.
3. Fit robust regression $DOY \sim year$ over the calibration window.
4. Predict past and future trends.
5. Draw ribbon (IQR), annual means, trend lines, climate windows.
6. Facet by country or by phase.

The robust regression uses `robustbase::lmrob`, which is resistant to outliers and non-normality, and therefore ideal for phenological time series.

Value

A `ggplot` object.

Author(s)

Matthias Templ

Examples

```
pep <- pep_download()
plot_phenology_trends(
  pep,
  species_name = "Vitis vinifera",
  phases = c(65, 81),
  layout = "country_phase"
)
```

`print.pep` *Print Method for PEP725 Data*

Description

Displays a concise summary of the PEP725 phenological dataset.

Usage

```
## S3 method for class 'pep'
print(x, n = 5, ...)
```

Arguments

`x`	A pep object.
`n`	Number of rows to display (default 5).
`...`	Additional arguments passed to print.

Value

Invisibly returns `x`.

Author(s)

Matthias Templ

Examples

```
pep <- pep_download()
print(pep)
```

print.pep_completeness *Print Method for Completeness Assessment*

Description

Print Method for Completeness Assessment

Usage

```
## S3 method for class 'pep_completeness'
print(x, n = 15, ...)
```

Arguments

`x`	A `pep_completeness` object
`n`	Number of rows to display
`...`	Additional arguments (unused)

Value

Invisibly returns `x`

Author(s)

Matthias Templ

print.pep_coverage *Print Method for PEP Coverage*

Description

Print Method for PEP Coverage

Usage

```
## S3 method for class 'pep_coverage'
print(x, ...)
```

Arguments

`x`	A `pep_coverage` object.
`...`	Additional arguments (unused).

Value

Invisibly returns `x`.

Author(s)

Matthias Templ

`print.pep_outliers` *Print Method for Outlier Detection Results*

Description

Print Method for Outlier Detection Results

Usage

```
## S3 method for class 'pep_outliers'
print(x, ...)
```

Arguments

`x`	A `pep_outliers` object
`...`	Additional arguments (unused)

Value

Invisibly returns `x`

`print.pep_quality` *Print Method for Data Quality Assessment*

Description

Print Method for Data Quality Assessment

Usage

```
## S3 method for class 'pep_quality'
print(x, n = 10, ...)
```

Arguments

`x`	A `pep_quality` object
`n`	Number of rows to display
`...`	Additional arguments (unused)

Value

Invisibly returns x

Author(s)

Matthias Templ

---

print.phase_check *Print Method for Phase Check Results*

Description

Print Method for Phase Check Results

Usage

```
## S3 method for class 'phase_check'
print(x, ...)
```

Arguments

x	A phase_check object
...	Additional arguments (unused)

Value

Invisibly returns x

Author(s)

Matthias Templ

---

print.phase_check_multi *Print Method for Multi-Species Phase Check*

Description

Print Method for Multi-Species Phase Check

Usage

```
## S3 method for class 'phase_check_multi'
print(x, ...)
```

Arguments

`x`	A `phase_check_multi` object
`...`	Additional arguments (unused)

Value

Invisibly returns `x`

Author(s)

Matthias Templ

`print.pheno_anomaly` *Print Method for Phenological Anomalies*

Description

Print Method for Phenological Anomalies

Usage

```
## S3 method for class 'pheno_anomaly'
print(x, n = 10, ...)
```

Arguments

`x`	A `pheno_anomaly` object
`n`	Number of rows to display
`...`	Additional arguments (unused)

Value

Invisibly returns `x`

Author(s)

Matthias Templ

`print.pheno_combined` *Print Method for Combined Time Series*

Description

Print Method for Combined Time Series

Usage

```
## S3 method for class 'pheno_combined'
print(x, ...)
```

Arguments

`x`	A `pheno_combined` object
`...`	Additional arguments (unused)

Value

Invisibly returns `x`

`print.pheno_gradient` *Print Method for Phenological Gradient Analysis*

Description

Print Method for Phenological Gradient Analysis

Usage

```
## S3 method for class 'pheno_gradient'
print(x, ...)
```

Arguments

`x`	A `pheno_gradient` object
`...`	Additional arguments (unused)

Value

Invisibly returns `x`

Author(s)

Matthias Templ

print.pheno_normals *Print Method for Phenological Normals*

Description

Print Method for Phenological Normals

Usage

```
## S3 method for class 'pheno_normals'
print(x, n = 10, ...)
```

Arguments

x	A pheno_normals object
n	Number of rows to display
...	Additional arguments (unused)

Value

Invisibly returns x

Author(s)

Matthias Templ

print.pheno_pls *Print Method for PLS Phenology Results*

Description

Print Method for PLS Phenology Results

Usage

```
## S3 method for class 'pheno_pls'
print(x, ...)
```

Arguments

x	A pheno_pls object
...	Additional arguments (unused)

Value

Invisibly returns x

`print.pheno_synchrony` *Print Method for Phenological Synchrony Analysis*

Description

Print Method for Phenological Synchrony Analysis

Usage

```
## S3 method for class 'pheno_synchrony'
print(x, n = 10, ...)
```

Arguments

`x`	A `pheno_synchrony` object
`n`	Number of rows to display
`...`	Additional arguments (unused)

Value

Invisibly returns `x`

Author(s)

Matthias Templ

`print.pheno_turning` *Print Method for Trend Turning Analysis*

Description

Print Method for Trend Turning Analysis

Usage

```
## S3 method for class 'pheno_turning'
print(x, ...)
```

Arguments

`x`	A `pheno_turning` object
`...`	Additional arguments (unused)

Value

Invisibly returns `x`

`print.second_events` *Print Method for Second Events Detection*

Description

Print Method for Second Events Detection

Usage

```
## S3 method for class 'second_events'
print(x, ...)
```

Arguments

`x`	A `second_events` object
`...`	Additional arguments (unused)

Value

Invisibly returns `x`

`select_phase` *Select and Label Phenophases from PEP Time Series*

Description

Extracts a subset of phenological observations from a processed PEP time series (e.g., output from species-level aggregation), filters by species, year, and phenophase ID, and attaches meaningful phase labels based on standard BBCH codes.

Usage

```
select_phase(dt, label, sp, yrmin, phases = c(60, 100))
```

Arguments

`dt`	A `data.table` containing columns `species`, `year`, `phase_id`, and `mean_day`. Typically produced by aggregating PEP725 data (e.g., via `pheno_regional`).
`label`	A character string used as identifier for source and site columns in the output.
`sp`	Species name (as character string) to filter, e.g., `"Triticum aestivum"`.
`yrmin`	Minimum year (inclusive) for filtering.
`phases`	Integer vector of phase IDs to include (default: `c(60, 100)` corresponding to "Heading" and "Harvest").

Details

This function uses an internal lookup table to map numeric `phase_id` codes to descriptive BBCH phase names (e.g., `60` = `"Heading"`, `100` = `"Harvest"`). Unmapped phase IDs will trigger a warning. If expected phases are missing from the result, a separate warning is issued.

The function is particularly useful for visualizing specific growth stages or comparing phenological trends across datasets and locations.

Value

A `data.table` with columns:

`year`	Year of observation
`phase`	Mapped BBCH phenological stage name
`DOY`	Mean day-of-year for this phase, species, and year
`source`	Set to `label`
`site`	Set to `label`

Author(s)

Matthias Templ

See Also

`pheno_regional`, `pep_download`, `pheno_plot`

Examples

```
pep <- pep_download()
agg <- pheno_regional(pep, species_name = "Triticum aestivum", phase = 60)$pep_agg
df <- select_phase(agg, label = "PEP725", sp = "Triticum aestivum", yrmin = 1961)
head(df)
```

`subspecies_report` *Comprehensive Subspecies Availability Report*

Description

This function produces a complete report for phenological subspecies availability, combining: (1) a tidy summary table, (2) a publication-ready wide table, and (3) a ggplot2 heatmap showing data presence/absence.

Usage

```
subspecies_report(
  pep,
  genus_name = NULL,
  species_name = NULL,
  subspecies_name = NULL,
  phases = c(65, 87),
  years = 1961:2024,
  subregions = c("Austria", "Germany-North", "Germany-South", "Switzerland"),
  include_empty = TRUE,
  metric = "all",
  bbch_names = NULL,
  return_results = TRUE
)
```

Arguments

`pep`	A PEP725-style data frame containing at least: `genus`, `species`, `subspecies`, `country`, `year`, `phase_id`, `s_id`, and `day` or `DOY`.
`genus_name`	Character (optional). Genus to filter.
`species_name`	Character (optional). Species to filter.
`subspecies_name`	Character vector (optional). One or more subspecies.
`phases`	Integer vector. BBCH phases to evaluate.
`years`	Numeric vector. Years to include.
`subregions`	Character vector of region/country names.
`include_empty`	Logical. If TRUE, include subspecies x country rows even when no observations are present.
`metric`	Character vector. Metrics to compute. Use `"all"` for: `"n_obs"`, `"n_stations"`, `"first_year"`, `"last_year"`, `"median_doy"`, `"completeness"`.
`bbch_names`	Optional named list mapping BBCH phase numbers to labels, e.g. `list("65" = "Flowering", "87" = "Maturity")`.
`return_results`	Logical. If TRUE, returns a list invisibly.

Details

Users may specify a **genus**, **species**, or one or more **subspecies**. The report evaluates BBCH phases, regions, years, completeness, and several optional availability metrics.

Value

Invisibly returns a list with:

summary_table Tidy long-format summary

wide_table Publication-ready wide-format summary

heatmap A ggplot2 object

Author(s)

Matthias Templ

See Also

`pheno_trend_turning` for trend visualization

Examples

```
pep <- pep_download()

# Use Alpine subset for faster computation
pep_alpine <- pep[country %in% c("Switzerland", "Austria")]

# Grapevine subspecies (wine varieties) - longest historical records
if (nrow(pep_alpine[species == "Vitis vinifera"]) > 0) {
  subspecies_report(
    pep_alpine,
    subspecies_name = c("Vitis vinifera Riesling",
                        "Vitis vinifera MUELLER THURGAU WEISS"),
    subregions = c("Switzerland", "Austria"),
    metric = "all",
    bbch_names = list("65"="Flowering", "81"="Veraison")
  )

  subspecies_report(
    pep_alpine,
    species_name = "Vitis vinifera",
    subregions = c("Switzerland", "Austria"),
    metric = c("n_obs","median_doy","completeness")
  )

  subspecies_report(
    pep_alpine,
    genus_name = "Vitis",
    subregions = c("Switzerland", "Austria"),
    metric = "all"
  )
}
```

`summarize_subspecies_availability`

Summarize Subspecies Phenological Data Availability

Description

Computes data availability metrics for specified genus, species, or subspecies across BBCH phases and regions. Metrics include numbers of observations, station counts, first/last observation years, median DOY, and completeness ratios.

Usage

```
summarize_subspecies_availability(
  pep,
  genus_name = NULL,
  species_name = NULL,
  subspecies_name = NULL,
  phases = c(65, 87),
  years = 1961:2024,
  subregions = c("Austria", "Germany-North", "Germany-South", "Switzerland"),
  include_empty = FALSE,
  metric = c("n_obs", "n_stations", "first_year", "last_year", "median_doy",
    "completeness")
)
```

Arguments

`pep`	A PEP725-style data frame containing at least: `genus`, `species`, `subspecies`, `country`, `year`, `phase_id`, `s_id`, and `day` or `DOY`.
`genus_name`	Character (optional). Genus to filter.
`species_name`	Character (optional). Species to filter.
`subspecies_name`	Character vector (optional). One or more subspecies.
`phases`	Integer vector. BBCH phases to evaluate.
`years`	Numeric vector. Years to include.
`subregions`	Character vector of region/country names.
`include_empty`	Logical. If TRUE, include subspecies x country rows even when no observations are present.
`metric`	Character vector specifying which metrics to compute. Allowed values: `"n_obs"`, `"n_stations"`, `"first_year"`, `"last_year"`, `"median_doy"`, `"completeness"`, or `"all"`.

Value

A tibble in long format with one row per `subspecies x country`.

Author(s)

Matthias Templ

Examples

```
pep <- pep_download()
pep_alpine <- pep[country %in% c("Switzerland", "Austria")]

# Grapevine species (longest historical records)
if (nrow(pep_alpine[species == "Vitis vinifera"]) > 0) {
  summarize_subspecies_availability(
    pep_alpine,
    species_name = "Vitis vinifera",
    subregions = c("Switzerland", "Austria"),
    metric = "all"
  )

  summarize_subspecies_availability(
    pep_alpine,
    subspecies_name = c("Vitis vinifera Riesling",
                        "Vitis vinifera MUELLER THURGAU WEISS"),
    subregions = c("Switzerland", "Austria"),
    metric = c("n_obs","median_doy")
  )
}
```

summary.pep *Summary Method for PEP725 Data*

Description

Provides a detailed phenological summary of the dataset, including breakdowns by species, phase, country, and temporal coverage.

Usage

```
## S3 method for class 'pep'
summary(object, by = c("species", "phase", "country", "year"), top = 10, ...)
```

Arguments

`object`	A pep object.
`by`	Character. Summarize by `"species"`, `"phase"`, `"country"`, or `"year"`. Default is `"species"`.
`top`	Integer. Number of top entries to show (default 10).
`...`	Additional arguments (currently unused).

Value

A `pep_summary` object (list) containing summary tables, printed to console.

Author(s)

Matthias Templ

Examples

```
pep <- pep_download()
summary(pep)
summary(pep, by = "phase")
summary(pep, by = "country")
```

`summary.pep_completeness` *Summary Method for Completeness Assessment*

Description

Summary Method for Completeness Assessment

Usage

```
## S3 method for class 'pep_completeness'
summary(object, ...)
```

Arguments

`object`	A `pep_completeness` object
`...`	Additional arguments (unused)

Value

Invisibly returns a summary list

Author(s)

Matthias Templ

`summary.pep_outliers` *Summary Method for Outlier Detection Results*

Description

Summary Method for Outlier Detection Results

Usage

```
## S3 method for class 'pep_outliers'
summary(object, ...)
```

Arguments

`object`	A `pep_outliers` object
`...`	Additional arguments (unused)

Value

Invisibly returns a summary data.table

`summary.pep_quality` *Summary Method for Data Quality Assessment*

Description

Summary Method for Data Quality Assessment

Usage

```
## S3 method for class 'pep_quality'
summary(object, ...)
```

Arguments

`object`	A `pep_quality` object
`...`	Additional arguments (unused)

Value

Invisibly returns a summary list

Author(s)

Matthias Templ

`summary.pheno_anomaly` *Summary Method for Phenological Anomalies*

Description

Summary Method for Phenological Anomalies

Usage

```
## S3 method for class 'pheno_anomaly'
summary(object, ...)
```

Arguments

`object`	A `pheno_anomaly` object
`...`	Additional arguments (unused)

Value

Invisibly returns a summary list

Author(s)

Matthias Templ

`summary.pheno_normals` *Summary Method for Phenological Normals*

Description

Summary Method for Phenological Normals

Usage

```
## S3 method for class 'pheno_normals'
summary(object, ...)
```

Arguments

`object`	A `pheno_normals` object
`...`	Additional arguments (unused)

Value

Invisibly returns a summary list

Author(s)

Matthias Templ

`summary.pheno_pls` *Summary Method for PLS Phenology Results*

Description

Summary Method for PLS Phenology Results

Usage

```
## S3 method for class 'pheno_pls'
summary(object, vip_threshold = 0.8, ...)
```

Arguments

`object`	A `pheno_pls` object
`vip_threshold`	Numeric. VIP threshold for identifying important periods. Default 0.8.
`...`	Additional arguments (unused)

Value

Invisibly returns a summary data.frame

`summary.pheno_synchrony` *Summary Method for Phenological Synchrony Analysis*

Description

Summary Method for Phenological Synchrony Analysis

Usage

```
## S3 method for class 'pheno_synchrony'
summary(object, ...)
```

Arguments

`object`	A `pheno_synchrony` object
`...`	Additional arguments (unused)

Value

Invisibly returns a summary list

Author(s)

Matthias Templ

summary.second_events *Summary Method for Second Events Detection*

Description

Summary Method for Second Events Detection

Usage

```
## S3 method for class 'second_events'
summary(object, ...)
```

Arguments

`object`	A `second_events` object
`...`	Additional arguments (unused)

Value

The summary data.table (invisibly)

[.pep *Subset PEP725 Data While Preserving Class*

Description

Ensures that subsetting operations return a pep object when the result still has the required structure.

Usage

```
## S3 method for class 'pep'
x[...]
```

Arguments

`x`	A pep object.
`...`	Arguments passed to the data.table subset method.

Value

A pep object if structure is preserved, otherwise a data.table.

Author(s)

Matthias Templ

Index

∗ **datasets**
    pep_seed, 40
[.pep, 101

add_country, 4
add_daylength, 5
as.pep, 5

bbch_description, 6, *22*

calc_daylength, *5*, 7
calc_max_daylength, 8
calc_thermal_sum, 9, *12*
calc_thermal_units, *9, 10*, 10

fread, *32*

get_phenology_doys, 12

is.pep, 14

kendall_tau, 15, *54*

lmrob, *49*

new_pep, 16

pep, *32*, *42*
pep-class, 17
pep725_demo, 17
pep_cache_clear, 18
pep_cache_info, 19
pep_check_connectivity, 20, *47*
pep_check_phases, 21, *24*, *26*
pep_check_phases_multi, 23
pep_completeness, *22*, 24
pep_coverage, *26*, 26
pep_download, *18, 19*, 28, *41*, *44*, *56*, *65*, *92*
pep_flag_outliers, 29, *33, 34*, *39*
pep_import, *28, 29*, 31, *66*
pep_plot_outliers, 33, *39*
pep_quality, *18*, *22*, *26*, *31*, *34*, 35
pep_second_events, 37
pep_seed, 40
pep_simulate, *29*, *40, 41*, 41
pheno_anomaly, *18*, *36*, 42, *56*, *63*, *68*, *75*, *78*
pheno_combine, *20*, *31*, 45
pheno_gradient, 48, *70*
pheno_leaflet, 50
pheno_map, 52
pheno_normals, *18*, *36*, *44*, *47*, *49*, *54*, 54, *68*,
    70, *75*, *77, 78*
pheno_plot, 57, *65*, *92*
pheno_plot_hh, 58, *66*
pheno_plot_timeseries, 59
pheno_pls, 61
pheno_regional, *57, 58*, 63, *91, 92*
pheno_regional_hh, *58, 59*, 65
pheno_synchrony, *54*, 67
pheno_trend_turning, *47*, *63*, 69, *94*
plot.pep, 71
plot.pep_completeness, 71
plot.pep_coverage, 72
plot.pep_outliers, 73
plot.pep_quality, 73
plot.pheno_anomaly, 75
plot.pheno_combined, 76
plot.pheno_gradient, 77
plot.pheno_normals, 77
plot.pheno_pls, 78
plot.pheno_synchrony, 79
plot.pheno_turning, 79
plot.second_events, 80
plot_phenology_trends, 81
print.pep, 83
print.pep_completeness, 84
print.pep_coverage, 84
print.pep_outliers, 85
print.pep_quality, 85
print.phase_check, 86

print.phase_check_multi, 86
print.pheno_anomaly, 87
print.pheno_combined, 88
print.pheno_gradient, 88
print.pheno_normals, 89
print.pheno_pls, 89
print.pheno_synchrony, 90
print.pheno_turning, 90
print.second_events, 91

R_user_dir, *28*
rbindlist, *32*

select_phase, 91
subspecies_report, 92
summarize_subspecies_availability, 94
summary.pep, 96
summary.pep_completeness, 97
summary.pep_outliers, 97
summary.pep_quality, 98
summary.pheno_anomaly, 98
summary.pheno_normals, 99
summary.pheno_pls, 100
summary.pheno_synchrony, 100
summary.second_events, 101