Package 'AeRobiology'

May 6, 2026

Type Package

Title A Computational Tool for Aerobiological Data

Version 2.0.2

Author Jesus Rojo [aut],
Antonio Picornell [aut],
Jose Oteros [aut, cre]

Maintainer Jose Oteros <OterosJose@gmail.com>

Description Different tools for managing databases of airborne particles, elaborating the main calculations and visualization of results. In a first step, data are checked using tools for quality control and all missing gaps are completed. Then, the main parameters of the pollen season are calculated and represented graphically. Multiple graphical tools are available: pollen calendars, phenological plots, time series, tendencies, interactive plots, abundance plots...

License GPL-3

Depends R (>= 2.10)

Imports dplyr, writexl, ggvis, lubridate, plotly, ggplot2, tidyr, circular, scales, grDevices, zoo, grid, data.table

Encoding UTF-8

LazyData true

RoxygenNote 7.3.3

Suggests testthat, knitr, rmarkdown

VignetteBuilder knitr

NeedsCompilation no

Repository CRAN

Date/Publication 2026-02-24 19:10:06 UTC

Contents

`AeRobiology` `AeRobiology` *package*

Description

AeRobiology computational tool

Author(s)

Maintainer: Jose Oteros <OterosJose@gmail.com>

Authors:

- Jesus Rojo <Jesus.Rojo@uclm.es>
- Antonio Picornell <picornell@uma.es>

`analyse_trend` *Calculating and Plotting Trends of Pollen Data (summary plot).*

Description

Function to calculate the main seasonal indexes of the pollen season (*Start Date*, *Peak Date*, *End Date* and *Pollen Integral*). Trends analysis of the parameters over the seasons. Summary dot plot showing the distribution of the main seasonal indexes over the years.

Usage

```
analyse_trend(
  data,
  interpolation = TRUE,
  int.method = "lineal",
  export.plot = TRUE,
  export.format = "pdf",
  export.result = TRUE,
  method = "percentage",
  quantil = 0.75,
  significant = 0.05,
  split = TRUE,
  result = "table",
  ...
)
```

Arguments

`data`	A `data.frame` object. This `data.frame` should include a first column in format `Date` and the rest of columns in format `numeric` belonging to each pollen type by column.
`interpolation`	A `logical` value specifying if the visualization shows the gaps in the inputs data (`interpolation = FALSE`) or if an interpolation method is used for filling the gaps (`interpolation = TRUE`). By default, `interpolation = TRUE`.
`int.method`	A `character` string with the name of the interpolation method to be used. The implemented methods that may be used are: `"lineal"`, `"movingmean"`, `"tseries"` or `"spline"`. By default, `int.method = "lineal"`.
`export.plot`	A `logical` value specifying if a plot will be exported or not. If `FALSE` graphical results will only be displayed in the active graphics window. If `TRUE` graphical results will be displayed in the active graphics window and also one pdf/png file will be saved within the *plot_AeRobiology* directory automatically created in the working directory. By default, `export.plot = TRUE`.
`export.format`	A `character` string specifying the format selected to save the plot. The implemented formats that may be used are: `"pdf"` or `"png"`. By default, `export.format = "pdf"`.
`export.result`	A `logical` value. If `export.result = TRUE`, a table is exported with the extension *.xlsx*, in the directory *table_AeRobiology*. This table has the information about the `slope` *"beta coefficient of a lineal model using as predictor the year and as dependent variable one of the main pollen season indexes"*. The information is referred to the main pollen season indexes: *Start Date*, *Peak Date*, *End Date* and *Pollen Integral*.
`method`	A `character` string specifying the method applied to calculate the pollen season and the main seasonal parameters. The implemented methods that can be used are: `"percentage"`, `"logistic"`, `"moving"`, `"clinical"` or `"grains"`. By default, `method = "percentage"` (`perc = 95%`). A more detailed information about the different methods for defining the pollen season may be consulted in the function `calculate_ps`.

`quantil`	A `numeric` value (between 0 and 1) indicating the quantile of data to be displayed in the graphical output of the function. `quantil = 1` would show all the values, however a lower quantile will exclude the most extreme values of the sample. To split the parameters using a different sampling units (e.g. dates and pollen concentrations) can be used low vs high values of `quantil` argument (e.g. 0.5 vs 1). Also can be used an extra argument: `split`. By default, `quantil = 0.75`. `quantil` argument can only be applyed when `split = FALSE`.
`significant`	A `numeric` value indicating the significant level to be considered in the linear trends analysis. This *p* level is displayed in the graphical output of the function. By default, `significant = 0.05`.
`split`	A `logical` argument. If `split = TRUE`, the plot is separated in two according to the nature of the variables (i.e. dates or pollen concentrations). By default, `split = TRUE`.
`result`	A `character` object with the definition of the object to be produced by the function. If `result == "plot"`, the function returns a list of objects of class **ggplot2**; if `result == "table"`, the function returns a **data.frame**. By default, `result = "table"`.
`...`	Additional arguments for the function `calculate_ps` are also accepted.

Details

This function allows to study time series trends of the pollen season. Even though the package was originally designed to treat aeropalynological data, it can be used to study many other atmospheric components (e.g., bacteria in the air, fungi, insects ...) *(Buters et al., 2018; Oteros et al., 2019)*. The study of trends in pollen time series is a common approach to study the impact of climate change or other environmental factors on vegetation (Galan et al., 2016; Garcia_Mozo et al., 2016; Recio et al., 2018). This tool can also be useful for studying trends in other fields (Oteros et al., 2015).

Value

If `result == "plot"`, the function returns a list of objects of class **ggplot2**; if `result == "table"`, the function returns a **data.frame** with the hourly patterns. The plot is of the class **ggplot2** or a list of plots of the class **ggplot2** (depending on the argument `split`). This is a combined dot plot showing the trends (*slope* and *p* value) of the main seasonal features.
The object of the class `data.frame` has the information about the `slope` *(beta coefficient of a lineal model using as predictor the year and as dependent variable one of the main pollen season indexes)*. The information is referred to the main pollen season indexes: *Start Date*, *Peak Date*, *End Date* and *Pollen Integral*.

References

Buters, J. T. M., Antunes, C., Galveias, A., Bergmann, K. C., Thibaudon, M., Galan, C., ... & Oteros, J. (2018). Pollen and spore monitoring in the world. *Clinical and translational allergy*, 8(1), 9.

Galan, C., Alcazar, P., Oteros, J., Garcia_Mozo, H., Aira, M. J., Belmonte, J., ... & Perez_Badia, R. (2016). Airborne pollen trends in the Iberian Peninsula. *Science of the Total Environment*, 550, 53_59.

Garcia_Mozo, H., Oteros, J. A., & Galan, C. (2016). Impact of land cover changes and climate on the main airborne pollen types in Southern Spain. *Science of the Total Environment*, 548, 221_228.

Oteros, J., Garcia_Mozo, H., Botey, R., Mestre, A., & Galan, C. (2015). Variations in cereal crop phenology in Spain over the last twenty_six years (1986_2012). *Climatic Change*, 130(4), 545_558.

Oteros, J., Bartusel, E., Alessandrini, F., Nunez, A., Moreno, D. A., Behrendt, H., ... & Buters, J. (2019). Artemisia pollen is the main vector for airborne endotoxin. *Journal of Allergy and Clinical Immunology*.

Recio, M., Picornell, A., Trigo, M. M., Gharbi, D., Garcia_Sanchez, J., & Cabezudo, B. (2018). Intensity and temporality of airborne Quercus pollen in the southwest Mediterranean area: Correlation with meteorological and phenoclimatic variables, trends and possible adaptation to climate change. *Agricultural and Forest Meteorology*, 250, 308_318.

See Also

calculate_ps; plot_trend

Examples

```
data("munich_pollen")
analyse_trend(munich_pollen, interpolation = FALSE, export.result = FALSE, export.plot = FALSE)
```

---

calculate_ps *Estimation of the Main Parameters of the Pollen Season*

Description

Function to calculate the main parameters of the pollen season with regard to phenology and pollen intensity from a historical database of several pollen types. The function can use the most common methods in the definition of the pollen season.

Usage

```
calculate_ps(
  data,
  method = "percentage",
  th.day = 100,
  perc = 95,
  def.season = "natural",
  reduction = FALSE,
  red.level = 0.9,
  derivative = 5,
  man = 11,
  th.ma = 5,
  n.clinical = 5,
  window.clinical = 7,
  window.grains = 5,
  th.pollen = 10,
```

```
  th.sum = 100,
  type = "none",
  interpolation = TRUE,
  int.method = "lineal",
  maxdays = 30,
  result = "table",
  plot = TRUE,
  export.plot = FALSE,
  export.result = FALSE
)
```

Arguments

data	A data.frame object including the general database where calculation of the pollen season must be applied. This data.frame must include a first column in Date format and the rest of columns in numeric format belonging to each pollen type by column.
method	A character string specifying the method applied to calculate the pollen season and the main parameters. The implemented methods that can be used are: "percentage", "logistic", "moving", "clinical" or "grains". A more detailed information about the different methods for defining the pollen season may be consulted in **Details**.
th.day	A numeric value. The number of days whose pollen concentration is bigger than this threshold is calculated for each year and pollen type. This value will be obtained in the results of the function. The th.day argument will be 100 by default.
perc	A numeric value ranging 0_100. This argument is valid only for method = "percentage". This value represents the percentage of the total annual pollen included in the pollen season, removing (100_perc)/2% of the total pollen before and after of the pollen season. The perc argument will be 95 by default.
def.season	A character string specifying the method for selecting the best annual period to calculate the pollen season. The pollen seasons may occur within the natural year between two years. The implemented options that can be used are: "natural", "interannual" or "peak". The def.season argument will be "natural" by default. A more detailed information about the different methods for selecting the best annual period to calculate the pollen season may be consulted in **Details**.
reduction	A logical value. This argument is valid only for the "logistic" method. If FALSE the reduction of the pollen data is not applicable. If TRUE a reduction of the peaks above a certain level (red.level argument) will be carried out before the definition of the pollen season. The reduction argument will be FALSE by default. A more detailed information about the reduction process may be consulted in **Details**.
red.level	A numeric value ranging 0_1 specifying the percentile used as level to reduce the peaks of the pollen series before the definition of the pollen season. This argument is valid only for the "logistic" method. The red.level argument will be 0.90 by default, specifying the percentile 90.

`derivative` A `numeric` (`integer`) value belonging to options of `4`, `5` or `6` specifying the derivative that will be applied to calculate the asymptotes which determines the pollen season using the `"logistic"` method. This argument is valid only for the `"logistic"` method. The `derivative` argument will be `5` by default. More information may be consulted in **Details**.

`man` A `numeric` (`integer`) value specifying the order of the moving average applied to calculate the pollen season using the `"moving"` method. This argument is valid only for the `"moving"` method. The `man` argument will be `11` by default.

`th.ma` A `numeric` value specifying the threshold used for the `"moving"` method for defining the beginning and the end of the pollen season. This argument is valid only for the `"moving"` method. The `th.ma` argument will be `5` by default.

`n.clinical` A `numeric` (`integer`) value specifying the number of days which must exceed a given threshold (`th.pollen` argument) for defining the beginning and the end of the pollen season. This argument is valid only for the `"clinical"` method. The `n.clinical` argument will be `5` by default.

`window.clinical`
A `numeric` (`integer`) value specifying the time window during which the conditions must be evaluated for defining the beginning and the end of the pollen season using the `"clinical"` method. This argument is valid only for the `"clinical"` method. The `window.clinical` argument will be `7` by default.

`window.grains` A `numeric` (`integer`) value specifying the time window during which the conditions must be evaluated for defining the beginning and the end of the pollen season using the `"grains"` method. This argument is valid only for the `"grains"` method. The `window.grains` argument will be `5` by default.

`th.pollen` A `numeric` value specifying the threshold that must be exceeded during a given number of days (`n.clinical` or `window.grains` arguments) for defining the beginning and the end of the pollen season using the `"clinical"` or `"grains"` methods. This argument is valid only for the `"clinical"` or `"grains"` methods. The `th.pollen` argument will be `10` by default.

`th.sum` A `numeric` value specifying the pollen threshold that must be exceeded by the sum of daily pollen during a given number of days (`n.clinical` argument) exceeding a given daily threshold (`th.pollen` argument) for defining the beginning and the end of the pollen season using the `"clinical"` method. This argument is valid only for the `"clinical"` method. The `th.sum` argument will be `100` by default.

`type` A `character` string specifying the parameters considered according to a specific pollen type for calculating the pollen season using the `"clinical"` method. The implemented pollen types that may be used are: `"birch"`, `"grasses"`, `"cypress"`, `"olive"` or `"ragweed"`. As result for selecting any of these pollen types the parameters `n.clinical`, `window.clinical`, `th.pollen` and `th.sum` will be automatically adjusted for the `"clinical"` method. If no pollen types are specified (`type = "none"`), these parameters will be considered by the user. This argument is valid only for the `"clinical"` method. The `type` argument will be `"none"` by default.

`interpolation` A `logical` value. If `FALSE` the interpolation of the pollen data is not applicable. If `TRUE` an interpolation of the pollen series will be applied to complete the gaps

with no data before the calculation of the pollen season. The `interpolation` argument will be `TRUE` by default. A more detailed information about the interpolation method may be consulted in **Details**.

`int.method` A `character` string specifying the method selected to apply the interpolation method in order to complete the pollen series. The implemented methods that may be used are: `"lineal"`, `"movingmean"`, `"spline"` or `"tseries"`. The `int.method` argument will be `"lineal"` by default.

`maxdays` A `numeric` (`integer` value) specifying the maximum number of consecutive days with missing data that the algorithm is going to interpolate. If the gap is bigger than the argument value, the gap will not be interpolated. Not valid with `int.method = "tseries"`. The `maxdays` argument will be `30` by default.

`result` A `character` string specifying the output for the function. The implemented outputs that may be obtained are: `"table"` and `"list"`. The argument `result` will be `"table"` by default.

`plot` A `logical` value specifying if a set of plots based on the definition of the pollen season will be shown in the plot history or not. If `FALSE` graphical results will not be shown. If `TRUE` a set of plots will be shown in the plot history. The `plot` argument will be `TRUE` by default.

`export.plot` A `logical` value specifying if a set of plots based on the definition of the pollen season will be saved in the workspace. If `TRUE` a *pdf* file for each pollen type showing graphically the definition of the pollen season for each studied year will be saved within the *plot_AeRobiology* directory created in the working directory. The `plot` argument will be `FALSE` by default.

`export.result` A `logical` value specifying if a excel file including all parameters for the definition of the pollen season saved in the working directoty will be required or not. If `FALSE` the results will not exported. If `TRUE` the results will be exported as *xlsx* file including all parameters calculated from the definition of the pollen season within the *table_AeRobiology* directory created in the working directory. The `export.result` argument will be `FALSE` by default.

Details

This function allows to calculate the pollen season using five different methods which are described below. After calculating the start_date, end_date and peak_date for the pollen season all rest of parameters have been calculated as detailed in **Value** section.

- `"percentage"` method. This is a commonly used method for defining the pollen season based on the elimination of a certain percentage in the beginning and the end of the pollen season *(Nilsson and Persson, 1981; Andersen, 1991)*. For example if the pollen season is based on the 95% of the total annual pollen (`"perc" = 95`), the start_date of the pollen season is marked as the day in which 2.5% of the total pollen is registered and the end_date of the pollen season is marked as the day in which 97.5% of the total pollen is registered.
- `"logistic"` method. This method was developed by *Ribeiro et al. (2007)* and modified by *Cunha et al. (2015)*. It is based on fitting annually a non_linear logistic regression model to the daily accumulated curve for each pollen type. This logistic function and the different derivatives were considered to calculate the start_date and end_date of the pollen season, based on the asymptotes when pollen amounts are stabilized on the beginning and the end of

the accumulated curve. For more information about the method to see *Ribeiro et al. (2007)* and *Cunha et al. (2015)*. Three different derivatives may be used (`derivative` argument) `4`, `5` or `6` that represent from higher to lower restrictive criterion for defining the pollen season. This method may be complemented with an optional method for reduction the peaks values (`reduction = TRUE`), thus avoiding the effect of the great influence of extreme peaks. In this sense, peaks values will be cut below a certain level that the user may select based on a percentile analysis of peaks. For example, `red.level = 0.90` will cut all peaks above the percentile `90`.

- `"moving"` method. This method is proposed the first time by the authors of this package. The definition of the pollen season is based on the application of a moving average to the pollen series in order to obtain the general seasonality of the pollen curve avoiding the great variability of the daily fluctuations. Thus, the start_date and the end_date will be established when the curve of the moving average reaches a given pollen threshold (`th.ma` argument). Also the order of the moving average may be customized by the user (`man` argument). By default, `man = 11` and `th.ma = 5`.
- `"clinical"` method. This method was proposed by *Pfaar et al. (2017)*. It is based on the expert consensus in relation to pollen exposure and the relationship with allergic symptoms derived of the literature. Different periods may be defined by this method: the pollen season, the high pollen season and the high pollen days. The start_date and end_date of the pollen season were defined as a certain number of days (`n.clinical` argument) within a time window period (`window.clinical` argument) exceeding a certain pollen threshold (`th.pollen` argument) which summation is above a certain pollen sum (`th.sum` argument). All these parameters are established for each pollen type according to *Pfaar et al. (2017)* and using the `type` argument these parameters may be automatically adjusted for the specific pollen types (`"birch"`, `"grasses"`, `"cypress"`, `"olive"` or `"ragweed"`). Furthermore, the user may change all parameters to do a customized definition of the pollen season. The start_date and end_date of the high pollen season were defined as three consecutive days exceeding a certain pollen threshold (`th.day` argument). The number of high pollen days will also be calculated exceeding this pollen threshold (`th.day`). For more information about the method to see *Pfaar et al. (2017)*.
- `"grains"` method. This method was proposed by *Galan et al. (2001)* originally in olive pollen but after also applied in other pollen types. The start_date and end_date of the pollen season were defined as a certain number of days (`window.grains` argument) exceeding a certain pollen threshold (`th.pollen` argument). For more information about the method to see *Galan et al. (2001)*.

The pollen season of the species may occur during the natural year (Calendar year: from 1. January to 31. December) or the start_date and the end_date of the pollen season may happen in two different natural years (or calendar years). This consideration has been taken into account and in this package different method for defining the period for calculating the pollen season have been implemented. In this sense, the `def.season` argument has been incorporated in three options:

- `"natural"`: considering the pollination year as natural year from 1st January to 31th December for defining the start_dates and end_dates of the pollen season for each pollen types.
- `"interannual"`: considering the pollination year from 1st June to 31th May for defining the start_dates and end_dates of the pollen season for each pollen types.
- `"peak"`: considering a customized pollination year for each pollen types calculated as 6 previous months and 6 later months from the average peak_date.

Pollen time series frequently have different gaps with no data and this fact could be a problem for the calculation of specific methods for defining the pollen season even providing incorrect results. In this sense by default a linear interpolation will be carried out to complete these gaps before to define the pollen season (`interpolation = TRUE`). Additionally, the users may select other interpolation methods using the `int.method` argument, as `"lineal"`, `"movingmean"`, `"spline"` or `"tseries"`. For more information to see the `interpollen` function.

Value

This function returns different results:
`data.frame` when `result = "table"` including the main parameters of the pollen season with regard to phenology and pollen intensity as:

- **type**: pollen type
- **seasons**: year of the beginning of the season
- **st.dt**: start_date (date)
- **st.jd**: start_date (day of the year)
- **en.dt**: end_date (date)
- **en.jd**: end_date (day of the year)
- **ln.ps**: length of the season
- **sm.tt**: total sum
- **sm.ps**: pollen integral
- **pk.val**: peak value
- **pk.dt**: peak_date (date)
- **pk.jd**: peak_date (day of year)
- **ln.prpk**: length of the pre_peak period
- **sm.prpk**: pollen integral of the pre_peak period
- **ln.pspk**: length of the post_peak period
- **sm.pspk**: pollen integral of the post_peak period
- **daysth**: number of days with more than 100 pollen grains
- **st.dt.hs**: start_date of the High pollen season (date, only for clinical method)
- **st.jd.hs**: start_date of the High pollen season (day of the year, only for clinical method)
- **en.dt.hs**: end_date of the High pollen season (date, only for clinical method)
- **en.jd.hs**: end_date of the High pollen season (day of the year, only for clinical method)

`list` when `result = "list"` including the main parameters of the pollen season, one pollen type by element `plots` when `plot = TRUE` showing graphically the definition of the pollen season for each studied year in the plot history. If `export.result = TRUE` this `data.frame` will also be exported as *xlsx* file within the *table_AeRobiology* directory created in the working directory. If `export.result = FALSE` the results will also be showed as list object named `list.ps`.
If `export.plot = TRUE` a *pdf* file for each pollen type showing graphically the definition of the pollen season for each studied year will be saved within the *plot_AeRobiology* directory created in the working directory.

References

Andersen, T.B., 1991. A model to predict the beginning of the pollen season. *Grana*, 30(1), pp.269_275.

Cunha, M., Ribeiro, H., Costa, P. and Abreu, I., 2015. A comparative study of vineyard phenology and pollen metrics extracted from airborne pollen time series. *Aerobiologia*, 31(1), pp.45_56.

Galan, C., Garcia_Mozo, H., Carinanos, P., Alcazar, P. and Dominguez_Vilches, E., 2001. The role of temperature in the onset of the *Olea europaea* L. pollen season in southwestern Spain. *International Journal of Biometeorology*, 45(1), pp.8_12.

Nilsson, S. and Persson, S., 1981. Tree pollen spectra in the Stockholm region (Sweden), 1973_1980. *Grana*, 20(3), pp.179_182.

Pfaar, O., Bastl, K., Berger, U., Buters, J., Calderon, M.A., Clot, B., Darsow, U., Demoly, P., Durham, S.R., Galan, C., Gehrig, R., Gerth van Wijk, R., Jacobsen, L., Klimek, L., Sofiev, M., Thibaudon, M. and Bergmann, K.C., 2017. Defining pollen exposure times for clinical trials of allergen immunotherapy for pollen_induced rhinoconjunctivitis_an EAACI position paper. *Allergy*, 72(5), pp.713_722.

Ribeiro, H., Cunha, M. and Abreu, I., 2007. Definition of main pollen season using logistic model. *Annals of Agricultural and Environmental Medicine*, 14(2), pp.259_264.

See Also

`interpollen`, `plot_ps`

Examples

```
data("munich_pollen")
calculate_ps(munich_pollen, plot = TRUE, result = "table")
```

`credits` *A view of the creators*

Description

Credits

Usage

```
credits()
```

`interpollen` *Interpolation of Missing Data in a Pollen Database by Different Methods*

Description

Function to simultaneously replace all missing data of an historical database of several pollen types by using different methods of interpolation.

Usage

```
interpollen(
  data,
  method = "lineal",
  maxdays = 30,
  plot = TRUE,
  factor = 2,
  ndays = 3,
  spar = 0.5,
  data2 = NULL,
  data3 = NULL,
  data4 = NULL,
  data5 = NULL,
  mincorr = 0.6,
  result = "wide"
)
```

Arguments

`data`	A `data.frame` object including the general database where interpollation must be performed. This `data.frame` must include a first column in `Date` format and the rest of columns in `numeric` format. Each column must contain information of one pollen type. It is not necessary to insert missing gaps; the function will automatically detect them.
`method`	A `character` string specifying the method applied to calculate and generate the pollen missing data. The implemented methods that can be used are: `"lineal"`, `"movingmean"`, `"spline"`, `"tseries"` or `"neighbour"`. A more detailed information about the different methods may be consulted in Details. The `method` argument will be `"lineal"` by default.
`maxdays`	A `numeric (interger)` value specifying the maximum number of consecutive days with missing data that the algorithm is going to interpolate. If the gap is bigger than the argument value, the gap will not be interpolated. Not valid with `"tseries"` method. The `maxdays` argument will be `30` by default.
`plot`	A `logical` argument. If `TRUE`, graphical previews of the input database will be plot at the end of the interpolation process. All the interpolated gaps will be marked in red. The `plot` argument will be `TRUE` by default.

`factor` A `numeric (interger)` value bigger than `1`. Only valid if the `"movingmean"` method is chosen. The argument specifies the factor which will multiply the gap size to stablish the range of the moving mean that will fulfill the gap. A more detailed information about the selection of the factor may be consulted in Details. The argument `factor` will be `1` by default.

`ndays` A `numeric (interger)` value bigger than `1`. Only valid if the `"spline"` method is chosen. Specifies the number of days beyond each side of the gap which are used to perform the spline regression. The argument `ndays` will be `3` by default.

`spar` A `numeric (double)` value ranging `0_1` specifying the degree of smoothness of the spline regression adjustment. As smooth as the adjustment is, more data are considered as outliers for the spline regression. Only valid if the `"spline"` method is chosen. The argument `"spar"` will be `0.5` by default.

`data2`, `data3`, `data4`, `data5`
A `data.frame` object (each one) including database of a neighbour pollen station which will be used to interpolate missing data in the target station. Only valid if the "neighbour" method is chosen. This `data.frame` must include a first column in `Date` format and the rest of columns in `numeric` format belonging to each pollen type by column. It is not necessary to insert the missing gaps; the function will automatically detect them. The arguments will be `NULL` by default.

`mincorr` A `numeric (double)` value ranging `0_1`. It specifies the minimal correlation coefficient (Spearman correlations) that neighbour stations must have with the target station to be taken into account for the interpolation. Only valid if the `"neighbour"` method is chosen. The argument `"mincorr"` will be `0.6` by default.

`result` A `character` string specifying the format of the resulting `data.frame`. Only `"wide"` or `"long"`. The `result` argument will be `"wide"` by default.

Details

This function allows to interpolate missing data in a pollen database using 4 different methods which are described below. Interpolation for each pollen type will be automatically done for gaps smaller than the `"maxdays"` argument.

- `"lineal"` method. The interpolation will be carried out by tracing a straight line between the gap extremes.
- `"movingmean"` method. It calculates the moving mean of the pollen daily concentrations with a window size of the gap size multiplicated by the `factor` argument and replace the missing data with the moving mean for these days. It is a dynamic function and for each gap of the database, the window size of the moving mean changes depending of each gap size.
- `"spline"` method. The interpolation will be carried out by performing a spline regression with the previous and following days to the gap. The number of days of each side of the gap that will be taken into account for calculating the spline regression are specified by `ndays` argument. The smoothness of the adjustment of the spline regression can be specified by the `spar` argument.
- `"tseries"` method. The interpolation will be carried out by analysing the time series of pollen database. It performs a seasonal_trend decomposition based on LOESS (Cleveland

et al., 1990). The seasonality of the historical database is extracted and used to predict the missing data by performing a linear regression with the target year.

- `"neighbour"` method. Other near stations provided by the user are used to interpolate the missing data of the target station. First of all, a Spearman correlation is performed between the target station and the neighbour stations to discard the neighbour stations with a correlation coefficient smaller than `mincorr` value. For each gap, a linear regression is performed between the neighbour stations and the target stations to determine the equation which converts the pollen concentrations of the neighbour stations into the pollen concentration of the target station. Only neighbour stations without any missing data during the gap period are taken into account for each gap.

Value

This function returns different results:

- If `result = "wide"`, returns a `data.frame` including the original data and completed with the interpolated data.
- If `result = "long"`, returns a `data.frame` containing your data in long format (the first column for date, the second for pollen type, the third for concentration and an additional fourth column with `1` if this data has been interpolated or `0` if not).
- If `plot = TRUE`, plots for each year and pollen type with daily values are represented in the active graphic window. Interpolated values are marked in red. If `method` argument is `"tseries"`, the seasonality is also represented in grey.

References

Cleveland RB, Cleveland WS, McRae JE, Terpenning I (1990) STL: a seasonal_trend decomposition procedure based on loess. J Off Stat 6(1):3_33.

See Also

`ma`

Examples

```
data("munich_pollen")
interpollen(munich_pollen, method = "lineal", plot = FALSE)
```

---

iplot_abundance *Plot of the Relative Abundance of the Pollen Types*

Description

Generates a barplot based on the relative abundance (as percentage) in the air of the pollen types with respect to the total amounts

Usage

```
iplot_abundance(
  data,
  n.types = 15,
  y.start = NULL,
  y.end = NULL,
  interpolation = TRUE,
  int.method = "lineal",
  col.bar = "#E69F00",
  type.plot = "static",
  result = "plot",
  export.plot = FALSE,
  export.format = "pdf",
  exclude = NULL,
  ...
)
```

Arguments

`data`	A `data.frame` object including the general database where calculation of the pollen season must be applied. This `data.frame` must include a first column in `Date` format and the rest of columns in `numeric` format belonging to each pollen type by column.
`n.types`	A `numeric` (`integer`) value specifying the number of the most abundant pollen types that must be represented in the plot of the relative abundance. A more detailed information about the selection of the considered pollen types may be consulted in **Details**. The `n.types` argument will be `15` types by default.
`y.start`, `y.end`	A `numeric` (`integer`) value specifying the period selected to calculate relative abundances of the pollen types (`start year _ end year`). If `y.start` and `y.end` are not specified (`NULL`), the entire database will be used to generate the pollen calendar. The `y.start` and `y.end` arguments will be `NULL` by default.
`interpolation`	A `logical` value. If `FALSE` the interpolation of the pollen data is not applicable. If `TRUE` an interpolation of the pollen series will be applied to complete the gaps with no data before the calculation of the pollen season. The `interpolation` argument will be `TRUE` by default. A more detailed information about the interpolation method may be consulted in **Details**.
`int.method`	A `character` string specifying the method selected to apply the interpolation method in order to complete the pollen series. The implemented methods that may be used are: `"lineal"`, `"movingmean"`, `"spline"` or `"tseries"`. The `int.method` argument will be `"lineal"` by default.
`col.bar`	A `character` string specifying the color of the bars to generate the graph showing the relative abundances of the pollen types. The `color` argument will be `#E69F00` by default, but any color may be selected.
`type.plot`	A `character` string specifying the type of plot selected to show the plot showing the relative abundance of the pollen types. The implemented types that may be used are: `static` generates a static **ggplot** object and `dynamic` generates a dynamic **plotly** object.

`result` A `character` string specifying the output for the function. The implemented outputs that may be obtained are: `"plot"` and `"table"`. The argument `result` will be `"plot"` by default.

`export.plot` A `logical` value specifying if a plot saved in the working directory will be required or not. If `FALSE` graphical results will only be displayed in the active graphics window. If `TRUE` graphical results will be displayed in the active graphics window and also a *pdf* or *png* file (according to the `export.format` argument) will be saved within the *plot_AeRobiology* directory automatically created in the working directory. This argument is applicable only for `"static"` plots. The `export.plot` will be `FALSE` by default.

`export.format` A `character` string specifying the format selected to save the plot showing the relative abundance of the pollen types. The implemented formats that may be used are: `"pdf"` and `"png"`. This argument is applicable only for `"static"` plots. The `export.format` will be `"pdf"` by default.

`exclude` A `character` string vector with the names of the pollen types to be excluded from the plot.

`...` Other additional arguments may be used to customize the exportation of the plots using `"pdf"` or `"png"` files and therefore arguments from `pdf` and `png` functions (**grDevices** package) may be implemented. For example, for *pdf* files the user may custom the arguments: `width`, `height`, `family`, `title`, `fonts`, `paper`, `bg`, `fg`, `pointsize...`; and for *png* files the user may custom the arguments: `width`, `height`, `units`, `pointsize`, `bg`, `res...`

Details

This function allows to calculate the relative abundance of the pollen types in the air from a database and to display a barplot with the percentage representation of the main pollen types as the graph reported by *Rojo et al. (2016)*. This plot will be generated only for the specified number of the most abundant pollen types using the `n.types` argument by the user.

Pollen time series frequently have different gaps with no data and this fact could be a problem for the calculation of specific methods for defining the pollen season even providing incorrect results. In this sense by default a linear interpolation will be carried out to complete these gaps before to define the pollen season (`interpolation = TRUE`). Additionally, the users may select other interpolation methods using the `int.method` argument, as `"lineal"`, `"movingmean"`, `"spline"` or `"tseries"`. For more information to see the `interpollen` function.

Value

This function returns different results:

`plot` in the active graphics window displaying the pollen calendar generated by the user when `result = "plot"`. This `plot` may be included in an object by assignment operators.

`data.frame` including the yearly average pollen amounts for each pollen types used to generate the pollen of the relative abundance when `result = "table"`. This `data.frame` will be included in an object named `annual.sum.data`.

If `export.plot = FALSE` graphical results will only be displayed in the active graphics window as **ggplot** graph. Additional characteristics may be incorporated to the plot by `ggplot` syntax (see **ggplot2** package).

If `export.plot = TRUE` and `export.format = pdf` a *pdf* file with the plot will be saved within the *plot_AeRobiology* directory created in the working directory. This option is applicable only for `"static"` plots. Additional characteristics may be incorporated to the exportation as *pdf* file (see **grDevices** package).

If `export.plot = TRUE` and `export.format = png` a *png* file with the plot will be saved within the *plot_AeRobiology* directory created in the working directory. This option is applicable only for `"static"` plots. Additional characteristics may be incorporated to the exportation as *png* file (see **grDevices** package).

If `type.plot = dynamic` graphical results will be displayed in the active Viewer window as **plotly** graph. Additional characteristics may be incorporated to the plot by `plotly` syntax (see **plotly** package).

References

Rojo, J., Rapp, A., Lara, B., Sabariego, S., Fernandez_Gonzalez, F. and Perez_Badia, R., 2016. Characterisation of the airborne pollen spectrum in Guadalajara (central Spain) and estimation of the potential allergy risk. *Environmental Monitoring and Assessment*, 188(3), p.130.

See Also

`interpollen`

Examples

```
data("munich_pollen")
iplot_abundance (munich_pollen, interpolation = FALSE, export.plot = FALSE)
```

---

iplot_pheno *Phenological Plot*

Description

Generates a boxplot based on phenological parameters (start_dates and end_dates) that are calculated by the estimation of the main parameters of the pollen season

Usage

```
iplot_pheno(
  data,
  method = "percentage",
  n.types = 15,
  th.day = 100,
```

```
  perc = 95,
  def.season = "natural",
  reduction = FALSE,
  red.level = 0.9,
  derivative = 5,
  man = 11,
  th.ma = 5,
  n.clinical = 5,
  window.clinical = 7,
  window.grains = 5,
  th.pollen = 10,
  th.sum = 100,
  type = "none",
  interpolation = TRUE,
  int.method = "lineal",
  type.plot = "static",
  export.plot = FALSE,
  export.format = "pdf",
  ...
)
```

Arguments

data	A data.frame object including the general database where calculation of the pollen season must be applied in order to generate the phenological plot based on the start_dates and end_dates. This data.frame must include a first column in Date format and the rest of columns in numeric format belonging to each pollen type by column.
method	A character string specifying the method applied to calculate the pollen season and the main parameters. The implemented methods that can be used are: "percentage", "logistic", "moving", "clinical" or "grains". A more detailed information about the different methods for defining the pollen season may be consulted in calculate_ps function.
n.types	A numeric (integer) value specifying the number of the most abundant pollen types that must be represented in the pollen calendar. A more detailed information about the selection of the considered pollen types may be consulted in **Details**. The n.types argument will be 15 types by default.
th.day	A numeric value in order to calculate the number of days when this level is exceeded for each year and each pollen type. This value will be obtained in the results of the function. The th.day argument will be 100 by default.
perc	A numeric value ranging 0_100. This argument is valid only for method = "percentage". This value represents the percentage of the total annual pollen included in the pollen season, removing (100_perc)/2% of the total pollen before and after of the pollen season. The perc argument will be 95 by default.
def.season	A character string specifying the method for selecting the best annual period to calculate the pollen season. The pollen seasons may occur within the natural year or otherwise may occur between two years which determines the

best annual period considered. The implemented options that can be used are: "natural", "interannual" or "peak". The def.season argument will be "natural" by default. A more detailed information about the different methods for selecting the best annual period to calculate the pollen season may be consulted in calculate_ps function.

reduction A logical value. This argument is valid only for the "logistic" method. If FALSE the reduction of the pollen data is not applicable. If TRUE a reduction of the peaks above a certain level (red.level argument) will be carried out before the definition of the pollen season. The reduction argument will be FALSE by default. A more detailed information about the reduction process may be consulted in calculate_ps function.

red.level A numeric value ranging 0_1 specifying the percentile used as level to reduce the peaks of the pollen series before the definition of the pollen season. This argument is valid only for the "logistic" method. The red.level argument will be 0.90 by default, specifying the percentile 90.

derivative A numeric (integer) value belonging to options of 4, 5 or 6 specifying the derivative that will be applied to calculate the asymptotes which determines the pollen season using the "logistic" method. This argument is valid only for the "logistic" method. The derivative argument will be 5 by default.

man A numeric (integer) value specifying the order of the moving average applied to calculate the pollen season using the "moving" method. This argument is valid only for the "moving" method. The man argument will be 11 by default.

th.ma A numeric value specifying the threshold used for the "moving" method for defining the beginning and the end of the pollen season. This argument is valid only for the "moving" method. The th.ma argument will be 5 by default.

n.clinical A numeric (integer) value specifying the number of days which must exceed a given threshold (th.pollen argument) for defining the beginning and the end of the pollen season. This argument is valid only for the "clinical" method. The n.clinical argument will be 5 by default.

window.clinical
A numeric (integer) value specifying the time window during which the conditions must be evaluated for defining the beginning and the end of the pollen season using the "clinical" method. This argument is valid only for the "clinical" method. The window.clinical argument will be 7 by default.

window.grains A numeric (integer) value specifying the time window during which the conditions must be evaluated for defining the beginning and the end of the pollen season using the "grains" method. This argument is valid only for the "grains" method. The window.grains argument will be 5 by default.

th.pollen A numeric value specifying the threshold that must be exceeded during a given number of days (n.clinical or window.grains arguments) for defining the beginning and the end of the pollen season using the "clinical" or "grains" methods. This argument is valid only for the "clinical" or "grains" methods. The th.pollen argument will be 10 by default.

th.sum A numeric value specifying the pollen threshold that must be exceeded by the sum of daily pollen during a given number of days (n.clinical argument)

exceeding a given daily threshold (th.pollen argument) for defining the beginning and the end of the pollen season using the "clinical" method. This argument is valid only for the "clinical" method. The th.sum argument will be 100 by default.

type — A character string specifying the parameters considered according to a specific pollen type for calculating the pollen season using the "clinical" method. The implemented pollen types that may be used are: "birch", "grasses", "cypress", "olive" or "ragweed". As result for selecting any of these pollen types the parameters n.clinical, window.clinical, th.pollen and th.sum will be automatically adjusted for the "clinical" method. If no pollen types are specified (type = "none"), these parameters will be considered by the user. This argument is valid only for the "clinical" method. The type argument will be "none" by default.

interpolation — A logical value. If FALSE the interpolation of the pollen data is not applicable. If TRUE an interpolation of the pollen series will be applied to complete the gaps with no data before the calculation of the pollen season. The interpolation argument will be TRUE by default. A more detailed information about the interpolation method may be consulted in **Details**.

int.method — A character string specifying the method selected to apply the interpolation method in order to complete the pollen series. The implemented methods that may be used are: "lineal", "movingmean", "spline" or "tseries". The int.method argument will be "lineal" by default.

type.plot — A character string specifying the type of plot selected to show the phenological plot. The implemented types that may be used are: "static" generates a static **ggplot** object and "dynamic" generates a dynamic **plotly** object.

export.plot — A logical value specifying if a phenological plot saved in the working directory will be required or not. If FALSE graphical results will only be displayed in the active graphics window. If TRUE graphical results will be displayed in the active graphics window and also a *pdf* or *png* file (according to the export.format argument) will be saved within the *plot_AeRobiology* directory automatically created in the working directory. This argument is applicable only for "static" plots. The export.plot will be FALSE by default.

export.format — A character string specifying the format selected to save the phenological plot. The implemented formats that may be used are: "pdf" and "png". This argument is applicable only for "static" plots. The export.format will be "pdf" by default.

... — Other additional arguments may be used to customize the exportation of the plots using *pdf* or *png* files and therefore arguments from pdf and png functions (**grDevices** package) may be implemented. For example, for *pdf* files the user may custom the arguments: width, height, family, title, fonts, paper, bg, fg, pointsize...; and for *png* files the user may custom the arguments: width, height, units, pointsize, bg, res...

Details

This function allows to calculate the pollen season using five different methods which are described in calculate_ps function. After calculating the start_date and end_date for each pollen type and

each year a phenological plot will be generated using the boxplot approach where axis x represents the time (Day of the Year) and axis y includes the considered pollen types. The phenological plot will be generated only for the specified number of the most abundant pollen types using the `n.types` argument by the user. The implemented methods for defining the pollen season includes the most commonly used methodologies (*Nilsson and Persson, 1981*; *Andersen, 1991*; *Galan et al., 2001*; *Ribeiro et al., 2007*; *Cunha et al., 2015*, *Pfaar et al., 2017*) and a new implemented method (see `calculate_ps` function).
Pollen time series frequently have different gaps with no data and this fact could be a problem for the calculation of specific methods for defining the pollen season even providing incorrect results. In this sense by default a linear interpolation will be carried out to complete these gaps before to define the pollen season (`interpolation = TRUE`). Additionally, the users may select other interpolation methods using the `int.method` argument, as `"lineal"`, `"movingmean"`, `"spline"` or `"tseries"`. For more information to see the `interpollen` function.

Value

This function returns different results:
If `export.plot = FALSE` graphical results will only be displayed in the active graphics window as **ggplot** graph. Additional characteristics may be incorporated to the plot by `ggplot` syntax (see **ggplot2** package).
If `export.plot = TRUE` and `export.format = pdf` a *pdf* file of the phenological plot will be saved within the *plot_AeRobiology* directory created in the working directory. This option is applicable only for `"static"` plots. Additional characteristics may be incorporated to the exportation as *pdf* file (see **grDevices** package).
If `export.plot = TRUE` and `export.format = png` a *png* file of the phenological plot will be saved within the *plot_AeRobiology* directory created in the working directory. This option is applicable only for `"static"` plots. Additional characteristics may be incorporated to the exportation *png* file (see **grDevices** package).
If `type.plot = dynamic` graphical results will be displayed in the active Viewer window as **plotly** graph. Additional characteristics may be incorporated to the plot `plotly` syntax (see **plotly** package).

References

Andersen, T.B., 1991. A model to predict the beginning of the pollen season. *Grana*, 30(1), pp.269_275.

Cunha, M., Ribeiro, H., Costa, P. and Abreu, I., 2015. A comparative study of vineyard phenology and pollen metrics extracted from airborne pollen time series. *Aerobiologia*, 31(1), pp.45_56.

Galan, C., Garcia_Mozo, H., Carinanos, P., Alcazar, P. and Dominguez_Vilches, E., 2001. The role of temperature in the onset of the *Olea europaea* L. pollen season in southwestern Spain. *International Journal of Biometeorology*, 45(1), pp.8_12.

Nilsson, S. and Persson, S., 1981. Tree pollen spectra in the Stockholm region (Sweden), 1973_1980. *Grana*, 20(3), pp.179_182.

Pfaar, O., Bastl, K., Berger, U., Buters, J., Calderon, M.A., Clot, B., Darsow, U., Demoly, P., Durham, S.R., Galan, C., Gehrig, R., Gerth van Wijk, R., Jacobsen, L., Klimek, L., Sofiev, M., Thibaudon, M. and Bergmann, K.C., 2017. Defining pollen exposure times for clinical trials of allergen immunotherapy for pollen_induced rhinoconjunctivitis_an EAACI position paper. *Allergy*, 72(5), pp.713_722.

Ribeiro, H., Cunha, M. and Abreu, I., 2007. Definition of main pollen season using logistic model. *Annals of Agricultural and Environmental Medicine*, 14(2), pp.259_264.

See Also

calculate_ps, interpollen

Examples

```
data("munich_pollen")
iplot_pheno (munich_pollen, interpolation = FALSE)
```

---

iplot_pollen *Interactive Plotting Pollen Data (one season).*

Description

Function to plot the pollen data during one season. The plots are fully interactive.

Usage

```
iplot_pollen(data, year)
```

Arguments

data	A data.frame object. This data.frame should include a first column in format Date and the rest of columns in format numeric belonging to each pollen type by column.
year	An integer value specifying the year to display. This is a mandatory argument.

Value

An interactive plot of the class **ggvis**.

See Also

iplot_years

Examples

```
data("munich_pollen")
iplot_pollen(data = munich_pollen, year = 2012)
```

iplot_years *Interactive Plotting Pollen Data (one pollen type).*

Description

Function to plot the pollen data of a pollen type during several seasons. The plots are fully interactive.

Usage

```
iplot_years(data, pollen)
```

Arguments

data	A data.frame object. This data.frame should include a first column in format Date and the rest of columns in format numeric belonging to each pollen type by column.
pollen	A character string with the name of the particle to show. This character must match with the name of a column in the input database. This is a mandatory argument.

Value

An interactive plot of the class **ggvis**.

See Also

iplot_pollen

Examples

```
data("munich_pollen")
iplot_years(data = munich_pollen, pollen = "Betula")
```

ma *Moving Average Calculator*

Description

Function to calculate the moving average of a given numeric vector. The order of the moving average may be customized by the user (man argument).

Usage

```
ma(data, man = 10, warnings = FALSE)
```

Arguments

data	A numeric vector (e.g. a numeric column of a dataframe).
man	An integer value specifying the order of the moving average applied to the data. By default, man = 10.
warnings	A logical value specifying the show of warning messages. By default, warnings = FALSE.

Value

This function returns a vector with the moving average of the input data.

Examples

```
data("munich_pollen")
ma(data = munich_pollen$Betula, man = 10, warnings = FALSE)
```

---

munich_pollen *Pollen data of Munich (2010_2015)*

Description

A dataset containing information of daily concentrations of pollen in the atmosphere of Munich during the years 2010_2015. Pollen types included: *"Alnus"*, *"Betula"*, *"Taxus"*, *"Fraxinus"*, *"Poaceae"*, *"Quercus"*, *"Ulmus"* and *"Urtica"*.

Usage

```
munich_pollen
```

Format

Time series of daily pollen concentrations expressed as pollen grains / m3 of air.

Details

Data were obtained at Munich (Zentrum Allergie und Umwelt, ZAUM) using a Hirst_type volumetric pollen trap (Hirst, 1952) following the standard methodology (VDI4252_4_2016). Some gaps have been added to test some functions of the package (e.g. quality_control, interpollen).

The data were obtained by the research team of Prof. Jeroen Buters (Christine Weil & Ingrid Weichenmeier). We specially acknowledge this team and the Zentrum Allergie und Umwelt (ZAUM, directed by Prof. Carsten B. Schmidt_Weber) for their support.

Source

https://www.zaum-online.de/

References

Hirst, J.M., 1952. AN AUTOMATIC VOLUMETRIC SPORE TRAP. Ann. Appl. Biol. 39, 257_265.

VDI 4252_4. (2016). Bioaerosole und biologische Agenzien_Ermittlung von Pollen und Sporen in der Aussenluft unterVerwendung einer volumetrischen Methode fuer einMessnetz zu allergologischen Zwecken. VDI_Richtlinie4252 Blatt 4, Entwurf. VDI/DIN_Handbuch Reinhaltungder Luft, Band 1a: Beuth, Berlin

---

`plot_heathour` *Plotting hourly patterns with heatplot*

Description

Function to plot pollen data expressed in concentrations with time resolution higher than 1 day (e.g. hourly, bi-hourly concentrations). As heatplot.

Usage

```
plot_heathour(
  data,
  locations = FALSE,
  low.col = "blue",
  mid.col = "white",
  high.col = "red"
)
```

Arguments

`data`	A `data.frame` object with the structure `long`. Where the first two columns are `factors` indicating the `pollen` and the `location`. The 3 and 4 columns are `POSIXct`, showing the hour. Where the third column is the beguinning of the concentration `from` and the fourth column is the end time of the concentrtion data `to`. The fifth column shows the concentrations of the different pollen types as `numeric`. Please see the example 3-hourly data from the automatic pollen monitor BAA500 from Munich and Viechtach in Bavaria (Germany) `data("POMO_pollen")`, supplied by ePIN Network supported by the Bavarian Government.
`locations`	A `logical` object with the specification if the different locations will be displayed in the plot. Argument only used when `result == "plot"`. By default, `locations = FALSE`.
`low.col`	A `character` object with the specification of the color of the lowest value for the scale. By default, `low.col = "blue"`.
`mid.col`	A `character` object with the specification of the color of the medium value for the scale. By default, `mid.col = "white"`.
`high.col`	A `character` object with the specification object with the specification of the color of the highest value for the scale. By default, `high.col = "red"`.

Value

The function returns an object or a list of objects of class **ggplot2**.

References

Oteros, J., Pusch, G., Weichenmeier, I., Heimann, U., Mueller, R., Roeseler, S., ... & Buters, J. T. (2015). Automatic and online pollen monitoring. *International archives of allergy and immunology*, 167(3), 158-166.

Examples

```
data("POMO_pollen")
plot_heathour(POMO_pollen)
```

---

plot_hour *Plotting hourly patterns*

Description

Function to plot pollen data expressed in concentrations with time resolution higher than 1 day (e.g. hourly, bi-hourly concentrations).

Usage

```
plot_hour(data, result = "plot", locations = FALSE)
```

Arguments

data	A `data.frame` object with the structure `long`. Where the first two columns are `factors` indicating the `pollen` and the `location`. The 3 and 4 columns are `POSIXct`, showing the hour. Where the third column is the beguinning of the concentration `from` and the fourth column is the end time of the concentrtion data `to`. The fifth column shows the concentrations of the different pollen types as `numeric`. Please see the example 3-hourly data from the automatic pollen monitor BAA500 from Munich and Viechtach in Bavaria (Germany) `data("POMO_pollen")`, supplied by ePIN Network supported by the Bavarian Government.
result	A `character` object with the definition of the object to be produced by the function. If `result == "plot"`, the function returns a list of objects of class **ggplot2**; if `result == "table"`, the function returns a **data.frame** with the hourly patterns. By default, `result = "plot"`.
locations	A `logical` object with the specification if the different locations will be displayed in the plot. Argument only used when `result == "plot"`. By default, `locations = FALSE`.

Value

If `result == "plot"`, the function returns a list of objects of class **ggplot2**; if `result == "table"`, the function returns a **data.frame** with the hourly patterns.

References

Oteros, J., Pusch, G., Weichenmeier, I., Heimann, U., Mueller, R., Roeseler, S., ... & Buters, J. T. (2015). Automatic and online pollen monitoring. *International archives of allergy and immunology*, 167(3), 158-166.

Examples

```
data("POMO_pollen")
plot_hour(POMO_pollen, result="plot", locations = FALSE)
plot_hour(POMO_pollen, result="plot", locations = TRUE)
```

---

plot_normsummary *Plotting the Amplitude of Several Pollen Seasons.*

Description

Function to plot the pollen data amplitude during several seasons: daily average pollen concentration over the study period, maximum pollen concentration of each day over the study period and minimum pollen concentration of each day value over the study period. It is possible to plot the relative abundance per day and smoothing the pollen season by calculating a moving average.

Usage

```
plot_normsummary(
  data,
  pollen,
  mave = 1,
  normalized = FALSE,
  interpolation = TRUE,
  int.method = "lineal",
  export.plot = FALSE,
  export.format = "pdf",
  color.plot = "orange2",
  axisname = "Pollen grains / m3",
  ...
)
```

Arguments

`data`	A `data.frame` object. This `data.frame` should include a first column in format `Date` and the rest of columns in format `numeric` belonging to each pollen type by column.

pollen	A character string with the name of the particle to show. This character must match with the name of a column in the input database. This is a mandatory argument.
mave	An integer value specifying the order of the moving average applied to the data. By default, mave = 1.
normalized	A logical value specifying if the visualization shows real pollen data (normalized = FALSE) or the percentage of every day over the whole pollen season (normalized = TRUE). By default, normalized = FALSE.
interpolation	A logical value specifying if the visualization shows the gaps in the inputs data (interpolation = FALSE) or if an interpolation method is used for filling the gaps (interpolation = TRUE). By default, interpolation = TRUE.
int.method	A character string with the name of the interpolation method to be used. The implemented methods that may be used are: "lineal", "movingmean", "tseries" or "spline". By default, int.method = "lineal".
export.plot	A logical value specifying if a plot will be exported or not. If FALSE graphical results will only be displayed in the active graphics window. If TRUE graphical results will be displayed in the active graphics window and also one pdf/png file will be saved within the *plot_AeRobiology* directory automatically created in the working directory. By default, export.plot = FALSE.
export.format	A character string specifying the format selected to save the plot. The implemented formats that may be used are: "pdf" or "png". By default, export.format = "pdf".
color.plot	A character string. The argument defines the color to fill the plot. Will be "orange2" by default.
axisname	A character string specifying the title of the y axis. By default, axisname = "Pollen grains / m3".
...	Other additional arguments may be used to customize the exportation of the plots using "pdf" or "png" files and therefore arguments from functions pdf and png may be implemented. For example, for pdf files the user may custom the arguments: width, height, family, title, fonts, paper, bg, fg, pointsize...; and for png files the user may custom the arguments: width, height, units, pointsize, bg, res...

Value

This function returns plot of class **ggplot2**. User are able to customize the output as a **ggplot2** object.

See Also

calculate_ps; plot_summary

Examples

```
data("munich_pollen")
plot_normsummary(munich_pollen, pollen = "Betula", interpolation = FALSE, export.plot = FALSE)
```

plot_ps *Pollen Season Plot*

Description

Function to plot the main pollen season of a single pollen type.

Usage

```
plot_ps(
  data,
  pollen.type = NULL,
  year = NULL,
  days = 30,
  fill.col = "turquoise4",
  int.method = "lineal",
  axisname = expression(paste("Pollen grains / m"^"3")),
  ...
)
```

Arguments

data	A data.frame object including the general database where interpollation must be performed. This data.frame must include a first column in Date format and the rest of columns in numeric format. Each column must contain information of one pollen type. It is not necessary to insert missing gaps; the function will automatically detect them.
pollen.type	A character string specifying the name of the pollen type which will be plotted. The name must be exactly the same that appears in the column name. Mandatory argument with no default.
year	A numeric (interger) value specifying the season to be plotted. The season does not necessary fit a natural year. See calculate_ps for more details. Mandatory argument with no default.
days	A numeric (interger) specifying the number of days beyond each side of the main pollen season that will be represented. The days argument will be 30 by default.
fill.col	A character string specifying the name of the color to fill the main pollen season (Galan et al., 2017) in the plot. See ggplot function for more details. The fill.col argument will be "turquoise4" by default.
int.method	A character string specifying the method selected to apply the interpolation method in order to complete the pollen series. The implemented methods that may be used are: "lineal", "movingmean", "spline" or "tseries". See interpollen function for more details. The int.method argument will be "lineal" by default.

axisname	A character string or an expression specifying the y axis title of the plot. The axisname argument will be expression(paste("Pollen grains / m" ^ "3")) by default.
...	Other arguments passed on to the pollen season calculation as specified in calculate_ps function.

Details

plot_ps function is designed to easily plot the main pollen season (Galan et al., 2017). The pre_peak period and the post_peak period are marked with different color intensity in the graph. The user must choose a single pollen type and season to plot.

Value

The function returns an object of class ggplot with a graphical representation of the main pollen season of the selected pollen type. The pre_peak and post_peak periods are marked with different color intensity.

References

Galan, C., Ariatti, A., Bonini, M., Clot, B., Crouzy, B., Dahl, A., Fernandez_Gonzalez, D., Frenguelli, G., Gehrig, R., Isard, S., Levetin, E., Li, D.W., Mandrioli, P., Rogers, C.A., Thibaudon, M., Sauliene, I., Skjoth, C., Smith, M., Sofiev, M., 2017. Recommended terminology for aerobiological studies. Aerobiologia (Bologna). 293_295.

See Also

calculate_ps, interpollen, ggplot.

Examples

```
data("munich_pollen")
plot_ps(munich_pollen, year = 2013, pollen.type = "Betula")
```

---

plot_summary *Plotting Several Pollen Seasons.*

Description

Function to plot the pollen data during several seasons. Also plots the averaged pollen season over the study period. It is possible to plot the relative abundance per day and smoothing the pollen season by calculating a moving average.

Usage

```
plot_summary(
  data,
  pollen,
  mave = 1,
  normalized = FALSE,
  interpolation = TRUE,
  int.method = "lineal",
  export.plot = FALSE,
  export.format = "pdf",
  axisname = "Pollen grains / m3",
  ...
)
```

Arguments

data	A data.frame object. This data.frame should include a first column in format Date and the rest of columns in format numeric belonging to each pollen type by column.
pollen	A character string with the name of the particle to show. This character must match with the name of a column in the input database. This is a mandatory argument.
mave	An integer value specifying the order of the moving average applied to the data. By default, mave = 1.
normalized	A logical value specifying if the visualization shows real pollen data (normalized = FALSE) or the percentage of every day over the whole pollen season (normalized = TRUE). By default, normalized = FALSE.
interpolation	A logical value specifying if the visualization shows the gaps in the inputs data (interpolation = FALSE) or if an interpolation method is used for filling the gaps (interpolation = TRUE). By default, interpolation = TRUE.
int.method	A character string with the name of the interpolation method to be used. The implemented methods that may be used are: "lineal", "movingmean", "tseries" or "spline". By default, int.method = "lineal".
export.plot	A logical value specifying if a plot will be exported or not. If FALSE graphical results will only be displayed in the active graphics window. If TRUE graphical results will be displayed in the active graphics window and also one pdf/png file will be saved within the *plot_AeRobiology* directory automatically created in the working directory. By default, export.plot = FALSE.
export.format	A character string specifying the format selected to save the plot. The implemented formats that may be used are: "pdf" or "png". By default, export.format = "pdf".
axisname	A character string specifying the title of the y axis. By default, axisname = "Pollen grains / m3".
...	Other additional arguments may be used to customize the exportation of the plots using "pdf" or "png" files and therefore arguments from functions pdf

and png may be implemented. For example, for pdf files the user may custom the arguments: width, height, family, title, fonts, paper, bg, fg, pointsize...; and for png files the user may custom the arguments: width, height, units, pointsize, bg, res...

Details

This function allows to summarize the pollen season by a simple plot. Even though the package was originally designed to treat aeropalynological data, it can be used to study many other atmospheric components (e.g., bacteria in the air, fungi, insects ...) *(Buters et al., 2018; Oteros et al., 2019)*.

Value

This function returns plot of class **ggplot2**. User are able to customize the output as a **ggplot2** object.

References

Buters, J. T. M., Antunes, C., Galveias, A., Bergmann, K. C., Thibaudon, M., Galan, C. & Oteros, J. (2018). Pollen and spore monitoring in the world. *Clinical and translational allergy*, 8(1), 9.

Oteros, J., Bartusel, E., Alessandrini, F., Nunez, A., Moreno, D. A., Behrendt, H., ... & Buters, J. (2019). Artemisia pollen is the main vector for airborne endotoxin. *Journal of Allergy and Clinical Immunology*.

See Also

calculate_ps; plot_normsummary

Examples

```
data("munich_pollen")
plot_summary(munich_pollen, pollen = "Betula", export.plot = FALSE, interpolation = FALSE)
```

`plot_trend` *Calculating and Plotting Trends of Pollen Data.*

Description

Function to calculate the main seasonal indexes of the pollen season (*Start Date*, *Peak Date*, *End Date* and *Pollen Integral*). Trends analysis of the parameters over the seasons. Plots showing the distribution of the main seasonal indexes over the years.

Usage

```
plot_trend(
  data,
  base_dir = getwd(),
  table_dir = file.path(base_dir, "table_AeRobiology"),
  interpolation = TRUE,
  int.method = "lineal",
  export.plot = TRUE,
  export.format = "pdf",
  export.result = TRUE,
  method = "percentage",
  ...
)
```

Arguments

data	A data.frame object. This data.frame should include a first column in format Date and the rest of columns in format numeric belonging to each pollen type by column.
base_dir	Character. Base directory where output tables are stored.
table_dir	Character. Directory where tables are exported.
interpolation	A logical value specifying if the visualization shows the gaps in the inputs data (interpolation = FALSE) or if an interpolation method is used for filling the gaps (interpolation = TRUE). By default, interpolation = TRUE.
int.method	A character string with the name of the interpolation method to be used. The implemented methods that may be used are: "lineal", "movingmean", "tseries" or "spline". By default, int.method = "lineal".
export.plot	A logical value specifying if a plot will be exported or not. If FALSE graphical results will only be displayed in the active graphics window. If TRUE graphical results will be displayed in the active graphics window and also one pdf/png file will be saved within the *plot_AeRobiology* directory automatically created in the working directory. By default, export.plot = TRUE.
export.format	A character string specifying the format selected to save the plot. The implemented formats that may be used are: "pdf" or "png". By default, export.format = "pdf".
export.result	A logical value. If export.result = TRUE, a table is exported with the extension *.xlsx*, in the directory *table_AeRobiology*. This table has the information about the slope *"beta coefficient of a lineal model using as predictor the year and as dependent variable one of the main pollen season indexes"*. The information is referred to the main pollen season indexes: *Start Date*, *Peak Date*, *End Date* and *Pollen Integral*.
method	A character string specifying the method applied to calculate the pollen season and the main seasonal parameters. The implemented methods that can be used are: "percentage", "logistic", "moving", "clinical" or "grains". By default, method = "percentage" (perc = 95%). A more detailed information about the different methods for defining the pollen season may be consulted in the function calculate_ps.

... Additional arguments for the function calculate_ps are also accepted.

Value

This function returns several plots in the directory *plot_AeRobiology/trend_plots* with the extension *.pdf* or *.png*.Also produces an object of the class data.frame and export a table with the extension *.xlsx*, in the directory *table_AeRobiology*.
These tables have the information about the slope *(beta coefficient of a lineal model using as predictor the year and as dependent variable one of the main pollen season indexes)*. The information is referred to the main pollen season indexes: *Start Date*, *Peak Date*, *End Date* and *Pollen Integral*.

See Also

calculate_ps; analyse_trend

Examples

```
data("munich_pollen")
plot_trend(munich_pollen, interpolation = FALSE, export.plot = FALSE, export.result = TRUE)
```

---

pollen_calendar *Pollen Calendar by Different Methods from a Historical Pollen Database*

Description

Function to calculate the pollen calendar from a historical database of several pollen types and using the most commonly used methods in the generation of the pollen calendars in the aerobiology field.

Usage

```
pollen_calendar(
  data,
  method = "heatplot",
  n.types = 15,
  start.month = 1,
  y.start = NULL,
  y.end = NULL,
  perc1 = 80,
  perc2 = 99,
  th.pollen = 1,
  average.method = "avg_before",
  period = "daily",
  method.classes = "exponential",
  n.classes = 5,
  classes = c(25, 50, 100, 300),
  color = "green",
  interpolation = TRUE,
```

```
  int.method = "lineal",
  na.remove = TRUE,
  result = "plot",
  export.plot = FALSE,
  export.format = "pdf",
  legendname = "Pollen grains / m3",
  ...
)
```

Arguments

data	A data.frame object including the general database where calculation of the pollen calendar must be applied. This data.frame must include a first column in Date format and the rest of columns in numeric format belonging to each pollen type by column.
method	A character string specifying the method applied to calculate and generate the pollen calendar. The implemented methods that can be used are: "heatplot", "violinplot" or "phenological". A more detailed information about the different methods for defining the pollen season may be consulted in **Details**. The method argument will be heatplot by default.
n.types	A numeric (integer) value specifying the number of the most abundant pollen types that must be represented in the pollen calendar. A more detailed information about the selection of the considered pollen types may be consulted in **Details**. The n.types argument will be 15 types by default.
start.month	A numeric (integer) value ranging 1_12 specifying the number of the month (January_December) when the beginning of the pollen calendar must be considered. This argument is only applicable for the "heatplot" method with "daily" period, for the "phenological" method with "avg_before" average.method, and for the "violinplot" method, and the rest of methods only may be generated from the January (start.month = 1). The start.month argument will be 1 (month of January) by default.
y.start, y.end	A numeric (integer) value specifying the period selected to calculate the pollen calendar (start year _ end year). If y.start and y.end are not specified (NULL), the entire database will be used to generate the pollen calendar. The y.start and y.end arguments will be NULL by default.
perc1, perc2	A numeric value ranging 0_100. These arguments are valid only for the "phenological" method. These values represent the percentage of the total annual pollen included in the pollen season, removing (100_percentage)/2% of the total pollen before and after of the pollen season. Two percentages must be specified because of the definition of the "main pollination period" (perc1) and "early/late pollination" (perc2) based on the "phenological" method proposed by *Werchan et al. (2018)*. The perc1 argument will be 80 and perc2 argument will be 99 by default. A more detailed information about the phenological method to generate the pollen calendar may be consulted in **Details**.
th.pollen	A numeric value specifying the minimum threshold of the average pollen concentration which will be used to generate the pollen calendar. Days below this threshold will not be considered. For the "phenological" method this value

limits the "possible occurrence" period as proposed by *Werchan et al. (2018)*. The th.pollen argument will be 1 by default. A more detailed information about the methods to generate the pollen calendar may be consulted in *Details*.

average.method A character string specifying the moment of the application of the average. This argument is valid only for the "phenological" method. The implemented methods that can be used are: "avg_before" or "avg_after". "avg_before" produces the average to the daily concentrations and then pollen season are calculated for all pollen types, this method is recommended as it is a more concordant method with respect to the rest of implemented methods. Otherwise, "avg_after" determines the pollen season for all years and all pollen types, and then a circular average is calculated for the start_dates and end_dates. The average.method argument will be "avg_before" by default.

period A character string specifying the interval time considered to generate the pollen calendar. This argument is valid only for the "heatplot" method. The implemented periods that can be used are: "daily" or "weekly". "daily" selection produces a pollen calendar using daily averages during the year and "weekly" selection produces a pollen calendar using weekly averages during the year. The period argument will be "daily" by default.

method.classes A character string specifying the method to define the classes used for classifying the average pollen concentrations to generate the pollen calendar. This argument is valid only for the "heatplot" method. The implemented methods for defining classes are: "exponential" and "custom". The method.classes argument will be "exponential" by default. A more detailed information about the methods to classify the average pollen concentrations to generate the pollen calendar may be consulted in **Details**.

n.classes A numeric (integer) value specifying the number of classes that will be used for classifying the average pollen concentrations to generate the pollen calendar. This argument is valid only for the "heatplot" method and the classification by method.classes = "custom". The n.classes argument will be 5 by default. A more detailed information about the methods to classify the average pollen concentrations to generate the pollen calendar may be consulted in **Details**.

classes A numeric vector specifying the threshold established to define the different classes that will be used for classifying the average pollen concentrations to generate the pollen calendar. This argument is valid only for the "heatplot" method and the classification by method.classes = "custom". The classes argument will be c(25, 50, 100, 300) by default. The number of specified classes must be equal to n.classes _ 1 because of the maximum threshold will be automatically specified by the maximum value. A more detailed information about the methods to classify the average pollen concentrations to generate the pollen calendar may be consulted in **Details**.

color A character string specifying the color to generate the graph showing the pollen calendar. This argument is valid only for the "heatplot" method. The implemented color palettes to generate the pollen calendar are: "green", "red", "blue", "purple" or "black". The color argument will be "green" by default.

interpolation A logical value. If FALSE the interpolation of the pollen data is not applicable. If TRUE an interpolation of the pollen series will be applied to complete the gaps

before the calculation of the pollen calendar. The `interpolation` argument will be `TRUE` by default. A more detailed information about the interpolation method may be consulted in **Details**.

`int.method` A `character` string specifying the method selected to apply the interpolation method in order to complete the pollen series. The implemented methods that may be used are: `"lineal"`, `"movingmean"`, `"spline"` or `"tseries"`. The `int.method` argument will be `"lineal"` by default.

`na.remove` A `logical` value specifying if `na` values must be remove for the pollen calendar or not. `n.remove = TRUE` by default.

`result` A `character` string specifying the output for the function. The implemented outputs that may be obtained are: `"plot"` and `"table"`. The argument `result` will be `"plot"` by default.

`export.plot` A `logical` value specifying if a plot with the pollen calendar saved in the working directory will be required or not. If `FALSE` graphical results will only be displayed in the active graphics window. If `TRUE` graphical results will be displayed in the active graphics window and also a *pdf* file will be saved within the *plot_AeRobiology* directory automatically created in the working directory. The `export.plot` will be `FALSE` by default.

`export.format` A `character` string specifying the format selected to save the pollen calendar plot. The implemented formats that may be used are: `"pdf"` and `"png"`. The `export.format` will be `"pdf"` by default.

`legendname` A `character` string specifying the title of the legend. By default is `"Pollen / m3"`.

`...` Other additional arguments may be used to customize the exportation of the plots using `"pdf"` or `"png"` files and therefore arguments from *pdf* and *png* functions (**grDevices** package) may be implemented. For example, for *pdf* files the user may custom the arguments: `width`, `height`, `family`, `title`, `fonts`, `paper`, `bg`, `fg`, `pointsize...`; and for *png* files the user may custom the arguments: `width`, `height`, `units`, `pointsize`, `bg`, `res...`

Details

This function allows to calculate and generate the pollen calendar using three different methods which are described below. The pollen calendar will be calculated and generated only for the period specified by the user using the `y.start` and `y.end` arguments, and for the specified number of the most abundant pollen types using the `n.types` argument by the user. The most abundant pollen types will be selected according to the highest average annual amounts of pollen registered by the pollen types during the considered period.

- `"heatplot"` method. This pollen calendar is constructed based on the daily or weekly average of pollen concentrations, depending of the preferences of the user that may select `"daily"` or `"weekly"` as `period` argument. Then, these averages may be classified in different categories following different methods selected by the user according to the `method.classes` argument. If `method.classes = "exponential"` the user will apply the classification based on exponential classes proposed by *Stix and Ferreti (1974)*, which has been commonly used in aerobiology for the generation of pollen calendars. The classification based on exponential method considers 11 classes (`1_2`, `3_5`, `6_11`, `12_24`, `25_49`, `50_99`, `100_199`, `200_399`,

400_799, 800_1600, >1600). An example of this pollen calendar may be consulted in *Rojo et al. (2016)*. This method to design pollen calendars is an adaptation from the pollen calendar proposed by *Spieksma (1991)* who considered 10_day periods instead of daily or weekly periods. Otherwise, if method.classes = "custom" the user may customize the classification according to the number of classes selected (n.classes argument) and the thresholds of the pollen concentrations used to define the classes (classes argument). Average values below the level of the th.pollen argument will be removed from the pollen calendar.

- "phenological" method. This pollen calendar is based on phenological definition of the pollen season and adapted from the methodology proposed by *Werchan et al. (2018)*. After to obtain daily average pollen concentrations for the most abundant pollen types different pollination periods were calculated using the daily averages. The main pollination period was calculated based on the percentage defined by perc1 argument (selected by the user, 80% by default) of the annual total pollen. For example, if perc1 = 80 the beginning of the high season was marked when 10% of the annual value was reached and the end was selected when 90% was reached. In the case of the early/late pollination a total of the percentage defined by perc2 argument (selected by the user, 99% by default) of the annual total pollen will be registered during this period. For this kind of pollen calendar the th.pollen argument will define the "possible occurrence" period as adapted by *Werchan et al. (2018)*, considering the entire period between the first and the last day when this pollen level is reached. In an alternative way the average may be carried out after to define the pollen seasons using method_average = "avg_after" (instead of "avg_before" by default). "avg_after" determines the pollen season for all years and all pollen types, and then an average for circular data is calculated from the start_dates and end_dates.
- "violinplot" method. This pollen calendar is based on the pollen intensity and adapted from the pollen calendar published by *ORourke (1990)*. In first time the daily averages of the pollen concentrations are calculated and then these averages are represented using the *violin plot* graph. The shape of the *violin plot* display the pollen intensity of the pollen types in a relative way i.e. the values will be calculated as relative measurements regarding to the most abundant pollen type in annual amounts. Therefore, this pollen calendar shows a relative comparison between the pollen intensity of the pollen types but without scales and units. Average values below the level of the th.pollen argument will be removed from the pollen calendar.

Pollen time series frequently have different gaps with no data and this fact could be a problem for the calculation of specific methods for defining the pollen season even providing incorrect results. In this sense by default a linear interpolation will be carried out to complete these gaps before to generate the pollen calendar. For more information to see the interpollen function.

Value

This function returns different results:
plot in the active graphics window displaying the pollen calendar generated by the user when result = "plot". This plot may be included in an object by assignment operators.
data.frame including the daily or weekly average pollen concentrations (according to the selection of the user) used to generate the pollen calendar. This data.frame will be returned when result = "table".
If export.plot = TRUE this plot displaying the pollen calendar will also be exported as file within the *Plot_AeRobiology*" directory created in the working directory.
If export.plot = TRUE and export.format = pdf a *pdf* file of the pollen calendar will be saved within the *plot_AeRobiology* directory created in the working directory. Additional characteristics

may be incorporated to the exportation as *pdf* file (see **grDevices** package)
If `export.plot = TRUE` and `export.format = png` a *png* file of the pollen calendar will be saved within the *plot_AeRobiology* directory created in the working directory. Additional characteristics may be incorporated to the exportation as *png* file (see **grDevices** package).

References

ORourke, M.K., 1990. Comparative pollen calendars from Tucson, Arizona: Durhamvs. Burkard samplers. *Aerobiologia*, 6(2), p.136_140.

Rojo, J., Rapp, A., Lara, B., Sabariego, S., Fernandez_Gonzalez, F. and Perez_Badia, R., 2016. Characterisation of the airborne pollen spectrum in Guadalajara (central Spain) and estimation of the potential allergy risk. *Environmental Monitoring and Assessment*, 188(3), p.130.

Spieksma, F.T.M., 1991. *Regional European pollen calendars. Allergenic pollen and pollinosis in Europe*, pp.49_65.

Stix, E. and Ferretti, M.L., 1974. *Pollen calendars of three locations in Western Germany. Atlas European des Pollens Allergisants*, pp.85_94.

Werchan, M., Werchan, B. and Bergmann, K.C., 2018. German pollen calendar 4.0_update based on 2011_2016 pollen data. *Allergo Journal International*, 27, pp.69_71.

See Also

`interpollen`, `calculate_ps`

Examples

```
data("munich_pollen")
pollen_calendar(munich_pollen, method = "heatplot", interplation = FALSE)
```

POMO_pollen — *3-hourly pollen data of Munich and Viechtach (2018), obtained automatically by BAA500*

Description

A dataset containing information of 3-hourly concentrations of pollen in the atmosphere of Munich (DEBIED) and Viechtach (DEVIEC) during the year 2018. Pollen types included: *"Poaceae"* and *"Pinus"*.

Usage

```
POMO_pollen
```

Format

Time series of 3-hours pollen concentrations expressed as pollen grains / m3 of air.

Details

Data were obtained by the automatic pollen monitor BAA500 from Munich and Viechtach in Bavaria (Germany) `data("POMO_pollen")`, supplied by the public ePIN Network supported by the Bavarian Government. The ePIN Network was built by Das Bayerische Landesamt für Gesundheit und Lebensmittelsicherheit (LGL) in collaboration with Zentrum Allergie und Umwelt (ZAUM).

Source

https://www.lgl.bayern.de/

https://www.zaum-online.de/

References

Oteros, J., et al., ... & Buters, J. T. (2019). Building an automatic Pollen Monitoring Network (ePIN): Selection of optimal sites by clustering pollen stations.

Oteros, J., Pusch, G., Weichenmeier, I., Heimann, U., Mueller, R., Roeseler, S., ... & Buters, J. T. (2015). Automatic and online pollen monitoring.

Hirst, J.M., 1952. AN AUTOMATIC VOLUMETRIC SPORE TRAP. Ann. Appl. Biol. 39, 257_265.

VDI 4252_4. (2016). Bioaerosole und biologische Agenzien_Ermittlung von Pollen und Sporen in der Aussenluft unterVerwendung einer volumetrischen Methode fuer einMessnetz zu allergologischen Zwecken. VDI_Richtlinie4252 Blatt 4, Entwurf. VDI/DIN_Handbuch Reinhaltungder Luft, Band 1a: Beuth, Berlin

---

`quality_control` *Quality Control of a Pollen Database*

Description

Function to check the quality of an historical database of several pollen types.

Usage

```
quality_control(
  data,
  int.window = 2,
  perc.miss = 20,
  ps.method = "percentage",
  result = "plot",
  th.day = 100,
  perc = 95,
  def.season = "natural",
  reduction = FALSE,
  red.level = 0.9,
  derivative = 5,
```

```
  man = 11,
  th.ma = 5,
  n.clinical = 5,
  window.clinical = 7,
  window.grains = 5,
  th.pollen = 10,
  th.sum = 100,
  type = "none",
  int.method = "lineal",
  ...
)
```

Arguments

data	A data.frame object including the general database where quality must be checked. This data.frame must include a first column in Date format and the rest of columns in numeric format belonging to each pollen type by column. It is not necessary to insert the missing gaps; the function will automatically detect them.
int.window	A numeric (interger)value bigger or equal to 1. The argument specifies the number of days of each side of the start, peak or end date of the main pollen season which will be checked during the quality control. If any of these days has been interpolated, the current season will not pass the quality control. The int.window argument will be 2 by default.
perc.miss	A numeric (interger) value between 0 and 100. The argument specifies the maximal percentage of interpolated days which is allowed inside the main pollen season to pass the quality control. The perc.miss argument will be 20 by default.
ps.method	A character string specifying the method applied to calculate the pollen season and the main parameters. The implemented methods that can be used are: "percentage", "logistic", "moving", "clinical" or "grains". A more detailed information about the different methods for defining the pollen season may be consulted in calculate_ps function. The ps.method argument will be "percentage" by default.
result	A character string specifying the format of the results. Only "plot" or "table" available. If "plot", graphical resume of the quality control will be plotted. If "table", a data.frame will be created indicating the filters passed by each pollen type and season. Consult 'Return' for more information .The result argument will be "plot" by default.
th.day	See calculate_ps for more details.
perc	See calculate_ps for more details.
def.season	See calculate_ps for more details.
reduction	See calculate_ps for more details.
red.level	See calculate_ps for more details.
derivative	See calculate_ps for more details.
man	See calculate_ps for more details.

th.ma	See calculate_ps for more details.
n.clinical	See calculate_ps for more details.
window.clinical	See calculate_ps for more details.
window.grains	See calculate_ps for more details.
th.pollen	See calculate_ps for more details.
th.sum	See calculate_ps for more details.
type	See calculate_ps for more details.
int.method	See calculate_ps for more details.
...	Other arguments passed on to the pollen season calculation as specified in calculate_ps function.

Details

Quality control is a relevant topic for aerobiology (Oteros et al., 2013). This function is another approach to improve the quality control management in the field.
quality_control function checks the quality of the pollen data of each pollen type and season. The filters applied by the function are:

- If the main pollen season (Galan et al., 2017) cannot be calculated according to calculate_ps function minimal requirements (lack of data for these pollen type and year). Filter named "Complete" in the "quality_control" data.frame.
- If the start, end or peak date of the main pollen season has been interpolated or a day near to it (number of days specified by int.window argument). If a day near to these dates is missing, the selected date could not be the right one. Filters named "Start", "Peak" and "End" in the "quality_control" data.frame.
- The percentage of missing data inside the main pollen season. It calculates the number of days which have been interpolated by the algorithm and their percentage inside the main pollen season. If a high percentage of the main pollen season has been interpolated, the information of these season could not be reliable. Filter named "Comp.MPS" in the "quality_control" data.frame.

Value

This function can return different results:

- If result = "plot": Graphical resume of the Quality Control results showing the seasons of each pollen type and their quality (the risk assumed if they are included in further studies). The legend indicates the number of filter that have been unsuccessfully passed for each case. Object of class ggplot. For graphical customization, see ggplot function.
- If result = "table": data.frame with logical values for each pollen type and season. If TRUE, the filter has been successfully passed for this case. If FALSE, this case does not fit the minimal requirements of this filter.

References

Galan, C., Ariatti, A., Bonini, M., Clot, B., Crouzy, B., Dahl, A., Fernandez_Gonzalez, D., Frenguelli, G., Gehrig, R., Isard, S., Levetin, E., Li, D.W., Mandrioli, P., Rogers, C.A., Thibaudon, M., Sauliene, I., Skjoth, C., Smith, M., Sofiev, M., 2017. Recommended terminology for aerobiological studies. Aerobiologia (Bologna). 293_295.

Oteros, J., Galan, C., Alcazar, P., & Dominguez_Vilches, E. (2013). Quality control in bio_monitoring networks, Spanish Aerobiology Network. Science of the Total Environment, 443, 559_565.

See Also

calculate_ps, interpollen, ggplot, ggsave

Examples

```
data("munich_pollen")
quality_control(munich_pollen[,c(1:4)])
```

Index