Package ‘reddPrec’

April 24, 2025

Title Reconstruction of Daily Data - Precipitation

Version 3.0.0

Description Applies quality control to daily precipitation observations;
reconstructs the original series by estimating precipitation in missing values; and
creates gridded datasets of daily precipitation.

License GPL-3

Encoding UTF-8

Imports terra, stats, foreach, doParallel, reshape, qmap, BreakPoints,
Kendall, car, geosphere, gridExtra, lattice, pracma, xts, zoo,
e1071, neuralnet, randomForest, reshape2, xgboost

RoxygenNote 7.3.2

NeedsCompilation no

Author Roberto Serrano-Notivoli [aut, cre]
(<`https://orcid.org/0000-0001-7663-1202`>),
Adrian Huerta [aut] (<`https://orcid.org/0000-0002-2415-3402`>),
Abel Centella [ctb] (<`https://orcid.org/0000-0002-1536-6043`>)

Maintainer Roberto Serrano-Notivoli <`roberto.serrano@unizar.es`>

Repository CRAN

Date/Publication 2025-04-24 11:30:02 UTC

Contents

eqc_Plot *Enhanced quality control plots for daily precipitation*

Description

The function create a matrix plot of the enhanced quality control tests for daily precipitation

Usage

```
eqc_Plot(xts_obj)
```

Arguments

`xts_obj`	xts of a single time series

Details

Six plots are built based on daily precipitation: time series, truncation, time series (threshold = 5 mm), small gaps, precision and rounding patterns and weekly cycle. These provide a visual inspection (Hunziker et al., 2017) but also how the automatic application (Huerta et al., 2020) of the enhanced quality control is applied.

References

Hunziker, S., Gubler, S., Calle, J., Moreno, I., Andrade, M., Velarde, F., ... & Brönnimann, S. (2017). Identifying, attributing, and overcoming common data quality issues of manned station observations. https://doi.org/10.1002/joc.5037

Huerta, A., Serrano-Notivoli, R., & Brönnimann, S. (2024). SC-PREC4SA: A serially complete daily precipitation dataset for South America. https://doi.org/10.31223/X57D8R

Examples

```
## Not run:
set.seed(123)

# creating fake daily precipitation data
dates_t <- seq(as.Date("1980-01-01"), as.Date("2015-12-31"), by = "day")
prec <- round(rnorm(length(dates_t), mean = 1.2, sd = 6), 1)
prec[prec<0] <- 0
xts_obj <- xts::xts(prec, dates_t)
names(xts_obj) <- "prec"
```

```
# enhanced qc plots
eqc_Plot(xts_obj)

# it also work if there is some empty data (but not if all is NA)
xts_obj["1990/2010"] <- NA
eqc_Plot(xts_obj)

## End(Not run)
```

`eqc_PrecisionRounding` *Precision and rounding patterns of daily precipitation*

Description

The function determine the level (0, 1 or 2) of the precision and rounding patterns test

Usage

```
eqc_PrecisionRounding(
  xts_obj,
  lmn_yday = 365 * 80/100,
  ths = list(lv0 = c(percent = 70), lv1to2 = c(percent = 50))
)
```

Arguments

`xts_obj`	xts of a single time series
`lmn_yday`	numeric value of the minimum number of days to be considered a complete year. The default value is 365 * 80 / 100 days
`ths`	list. List of parameters to define in which level would be the daily precipitation.

Details

Precision and rounding patterns depict inconsistencies in the frequency of decimal values in the time series. As there is no absolute correct frequency of decimals, we decided to measure how similar the decimal patterns are in the time series. A decimal pattern is interpreted as the list of unique decimal values observed, sorted in descending order. In this way, the decimal pattern for each year is computed first, followed by the selection of the most dominating pattern (mode). Based on how much (in percentage) this dominating pattern represents the time series, it is defined: Level 0: coherent precision and rounding pattern (similar decimal pattern in more (or equal) than 70% of the time series). Level 1: a similar decimal pattern in less than 70% but more than (or equal) 50% of the time series. Level 2: different decimal patterns (no dominant pattern). Argument ths present the above default thresholds for the levels definition.

References

Hunziker, S., Gubler, S., Calle, J., Moreno, I., Andrade, M., Velarde, F., ... & Brönnimann, S. (2017). Identifying, attributing, and overcoming common data quality issues of manned station observations. https://doi.org/10.1002/joc.5037

Huerta, A., Serrano-Notivoli, R., & Brönnimann, S. (2024). SC-PREC4SA: A serially complete daily precipitation dataset for South America. https://doi.org/10.31223/X57D8R

Examples

```
## Not run:
set.seed(123)

# creating fake daily precipitation data
dates_t <- seq(as.Date("1980-01-01"), as.Date("2015-12-31"), by = "day")
prec <- round(rnorm(length(dates_t), mean = 1.2, sd = 6), 1)
prec[prec<0] <- 0
xts_obj <- xts::xts(prec, dates_t)
names(xts_obj) <- "prec"

# Comparison with visual inspection
eqc_Plot(xts_obj)
eqc_PrecisionRounding(xts_obj)

# it also work if there is some empty data (but not if all is NA)
xts_obj["1990/2010"] <- NA
eqc_Plot(xts_obj)
eqc_PrecisionRounding(xts_obj)

# creating some rounding values
xts_obj["1980/1990"] <- round(xts_obj["1980/1990"], 0)
eqc_Plot(xts_obj)
eqc_PrecisionRounding(xts_obj)


## End(Not run)
```

eqc_SmallGaps *Small gaps level of daily precipitation*

Description

The function determine the level (0, 1 or 2) of the small gaps test

Usage

```
eqc_SmallGaps(
  xts_obj,
  lmn_yday = 365 * 80/100,
  ths = list(lv0 = c(percent = 0), lv1to2 = c(percent = 20))
)
```

Arguments

`xts_obj`	xts of a single time series
`lmn_yday`	numeric value of the minimum number of days to be considered a complete year. The default value is 365 * 80 / 100 days
`ths`	list. List of parameters to define in which level would be the daily precipitation.

Details

Small gaps can be seen as unreported precipitation events that result in a gap or a frequency reduction in values below a specific threshold. To define the small gaps, we calculated the total count of values in five precipitation ranges from 0-1, 1-2, 2-3, 3-4, and 4-5 mm (not including the values in the limits) for each year. Therefore, considering the percentage of years with zero counts: Level 0: no small gaps (0%; years with at least one value in any of the precipitation ranges). Level 1: small gaps in at least 20% of years. Level 2: small gaps in more than 20% of years Argument ths present the above default thresholds for the levels definition.

References

Hunziker, S., Gubler, S., Calle, J., Moreno, I., Andrade, M., Velarde, F., ... & Brönnimann, S. (2017). Identifying, attributing, and overcoming common data quality issues of manned station observations. https://doi.org/10.1002/joc.5037

Huerta, A., Serrano-Notivoli, R., & Brönnimann, S. (2024). SC-PREC4SA: A serially complete daily precipitation dataset for South America. https://doi.org/10.31223/X57D8R

Examples

```
## Not run:
set.seed(123)

# creating fake daily precipitation data
dates_t <- seq(as.Date("1980-01-01"), as.Date("2015-12-31"), by = "day")
prec <- round(rnorm(length(dates_t), mean = 1.2, sd = 6), 1)
prec[prec<0] <- 0
prec[prec<3 & prec>2] <- 0
xts_obj <- xts::xts(prec, dates_t)
names(xts_obj) <- "prec"

# Comparison with visual inspection
eqc_Plot(xts_obj)
eqc_SmallGaps(xts_obj)

# it also work if there is some empty data (but not if all is NA)
xts_obj["1990/2010"] <- NA
eqc_Plot(xts_obj)
eqc_SmallGaps(xts_obj)
```

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

eqc_Truncation *Truncation level of daily precipitation*

Description

The function determine the level (0, 1 or 2) of the truncation test

Usage

```
eqc_Truncation(
  xts_obj,
  ths = list(lv0 = c(n_years = 2), lv1to2 = c(n_years_l = 3, n_years_h = 5))
)
```

Arguments

xts_obj	xts of a single time series
ths	list. List of parameters to define in which level would be the daily precipitation.

Details

Truncation is when heavy precipitation episodes are truncated or noticeably reduced in frequency above a given threshold. Because there is no preceding algorithm for truncation, it is defined here as when the maximum boundary of a time series lasts for a set length of time (years). The maximum boundary is computed as the daily precipitation's maximum moving window value. Thus, based on the length of years: Level 0: no truncation (a constant maximum boundary lasts at least 2 years). Level 1: a constant maximum boundary lasts longer equal (or above) 3 years but less than 5 years Level 2: a constant maximum boundary lasts more than 5 years. Argument ths present the above default thresholds for the levels definition.

References

Hunziker, S., Gubler, S., Calle, J., Moreno, I., Andrade, M., Velarde, F., ... & Brönnimann, S. (2017). Identifying, attributing, and overcoming common data quality issues of manned station observations. https://doi.org/10.1002/joc.5037

Huerta, A., Serrano-Notivoli, R., & Brönnimann, S. (2024). SC-PREC4SA: A serially complete daily precipitation dataset for South America. https://doi.org/10.31223/X57D8R

Examples

```
## Not run:
set.seed(123)

# creating fake daily precipitation data
dates_t <- seq(as.Date("1980-01-01"), as.Date("2015-12-31"), by = "day")
prec <- round(rnorm(length(dates_t), mean = 1.2, sd = 50), 1)
prec[prec<0] <- 0
prec[prec<3 & prec>2] <- 0
xts_obj <- xts::xts(prec, dates_t)
names(xts_obj) <- "prec"

# Comparison with visual inspection
eqc_Plot(xts_obj)
eqc_Truncation(xts_obj)

# it also work if there is some empty data (but not if all is NA)
xts_obj["1990/2010"] <- NA
eqc_Plot(xts_obj)
eqc_Truncation(xts_obj)


## End(Not run)
```

---

eqc_Ts *Automatic enhanced quality control for daily precipitation time series*

Description

The function determine the level (0, 1 or 2) of the enhanced quality control tests

Usage

```
eqc_Ts(
  prec,
  sts,
  lmn_yday = 365 * 80/100,
  ths_trc = list(lv0 = c(n_years = 2), lv1to2 = c(n_years_l = 3, n_years_h = 5)),
  ths_sgs = list(lv0 = c(percent = 0), lv1to2 = c(percent = 20)),
  ths_wcc = list(lv0 = c(n_days = 0), lv1to2 = c(n_days = 2, percent = 10)),
  ths_prp = list(lv0 = c(percent = 70), lv1to2 = c(percent = 50)),
  ncpu = 1
)
```

Arguments

`prec`	xts matrix of precipitation time series

`sts`	data.frame with metadata of the stations. A column "ID" (unique ID of stations) is required.
`lmn_yday`	numeric. value of the minimum number of days to be considered a complete year. The default value is 365 * 80 / 100 days
`ths_trc`	list. List of parameters to define in which level would be the daily precipitation for truncation.
`ths_sgs`	list. List of parameters to define in which level would be the daily precipitation for small gaps.
`ths_wcc`	list. List of parameters to define in which level would be the daily precipitation for weekly cycle.
`ths_prp`	list. List of parameters to define in which level would be the daily precipitation for precision and rounding patterns.
`ncpu`	integer. number of CPU threads to use for parallel computing.

Details

Truncation is when heavy precipitation episodes are truncated or noticeably reduced in frequency above a given threshold. Because there is no preceding algorithm for truncation, it is defined here as when the maximum boundary of a time series lasts for a set length of time (years). The maximum boundary is computed as the daily precipitation's maximum moving window value. Thus, based on the length of years: Level 0: no truncation (a constant maximum boundary lasts at least 2 years). Level 1: a constant maximum boundary lasts longer equal (or above) 3 years but less than 5 years Level 2: a constant maximum boundary lasts more than 5 years. Small gaps can be seen as unreported precipitation events that result in a gap or a frequency reduction in values below a specific threshold. To define the small gaps, we calculated the total count of values in five precipitation ranges from 0-1, 1-2, 2-3, 3-4, and 4-5 mm (not including the values in the limits) for each year. Therefore, considering the percentage of years with zero counts: Level 0: no small gaps (0%; years with at least one value in any of the precipitation ranges). Level 1: small gaps in at least 20% of years. Level 2: small gaps in more than 20% of years Weekly cycles are characterized by the occurrence of wet days that significantly differ between the days of the week. To compute the weekly cycles, first, for each day of the week, the probability of precipitation is calculated by dividing the total number of wet days by the total counts of values. Later, the number of wet days is tested by a two-sided binomial test (95 % confidence level). Based on how many days were significant, it was defined: Level 0: no atypical weekly cycle (similar probability between the days of the week). Level 1: at least two days present an atypical probability (significant test). Level 2: more than two days present an atypical probability (significant test) or one day presents an extremely different probability (more than 10%). Precision and rounding patterns depict inconsistencies in the frequency of decimal values in the time series. As there is no absolute correct frequency of decimals, we decided to measure how similar the decimal patterns are in the time series. A decimal pattern is interpreted as the list of unique decimal values observed, sorted in descending order. In this way, the decimal pattern for each year is computed first, followed by the selection of the most dominating pattern (mode). Based on how much (in percentage) this dominating pattern represents the time series, it is defined: Level 0: coherent precision and rounding pattern (similar decimal pattern in more (or equal) than 70% of the time series). Level 1: a similar decimal pattern in less than 70% but more than (or equal) 50% of the time series. Level 2: different decimal patterns (no dominant pattern).

References

Hunziker, S., Gubler, S., Calle, J., Moreno, I., Andrade, M., Velarde, F., ... & Brönnimann, S. (2017). Identifying, attributing, and overcoming common data quality issues of manned station observations. https://doi.org/10.1002/joc.5037

Huerta, A., Serrano-Notivoli, R., & Brönnimann, S. (2024). SC-PREC4SA: A serially complete daily precipitation dataset for South America. https://doi.org/10.31223/X57D8R

`eqc_WeeklyCycle` *Weekly cycle level of daily precipitation*

Description

The function determine the level (0, 1 or 2) of the weekly cycle test

Usage

```
eqc_WeeklyCycle(
  xts_obj,
  ths = list(lv0 = c(n_days = 0), lv1to2 = c(n_days = 2, percent = 10))
)
```

Arguments

`xts_obj`	xts of a single time series
`ths`	list. List of parameters to define in which level would be the daily precipitation.

Details

Weekly cycles are characterized by the occurrence of wet days that significantly differ between the days of the week. To compute the weekly cycles, first, for each day of the week, the probability of precipitation is calculated by dividing the total number of wet days by the total counts of values. Later, the number of wet days is tested by a two-sided binomial test (95 % confidence level). Based on how many days were significant, it was defined: Level 0: no atypical weekly cycle (similar probability between the days of the week). Level 1: at least two days present an atypical probability (significant test). Level 2: more than two days present an atypical probability (significant test) or one day presents an extremely different probability (more than 10%). Argument ths present the above default thresholds for the levels definition.

References

Hunziker, S., Gubler, S., Calle, J., Moreno, I., Andrade, M., Velarde, F., ... & Brönnimann, S. (2017). Identifying, attributing, and overcoming common data quality issues of manned station observations. https://doi.org/10.1002/joc.5037

Huerta, A., Serrano-Notivoli, R., & Brönnimann, S. (2024). SC-PREC4SA: A serially complete daily precipitation dataset for South America. https://doi.org/10.31223/X57D8R

Examples

```
## Not run:
set.seed(123)

# creating fake daily precipitation data
dates_t <- seq(as.Date("1980-01-01"), as.Date("2015-12-31"), by = "day")
prec <- round(rnorm(length(dates_t), mean = 1.2, sd = 6), 1)
prec[prec<0] <- 0
xts_obj <- xts::xts(prec, dates_t)
names(xts_obj) <- "prec"

# Comparison with visual inspection
eqc_Plot(xts_obj)
eqc_WeeklyCycle(xts_obj)

# it also work if there is some empty data (but not if all is NA)
xts_obj["1990/2010"] <- NA
eqc_Plot(xts_obj)
eqc_WeeklyCycle(xts_obj)


## End(Not run)
```

---

gapFilling — *Estimating new values in original missing values data series of daily precipitation*

Description

This function uses the neighboring observations to estimate new precipitation values in those days and locations where no records exist.

Usage

```
gapFilling(
  prec,
  sts,
  model_fun = learner_glm,
  dates,
  stmethod = NULL,
  thres = NA,
  neibs = 10,
  coords,
  crs,
  coords_as_preds = TRUE,
  window,
  ncpu = 2
)
```

Arguments

`prec`	matrix containing the original (cleaned) precipitation data. Each column represents one station. The names of columns must coincide with the names of the stations.
`sts`	data.frame. A column "ID" (unique ID of stations) is required. The rest of the columns (all of them) will act as predictors of the model.
`model_fun`	function. A function that integrates the statistical hybrid model (classification and regression). The default is learner_glm, which is the original model. Other models are also available (learner_rf and learner_xgboost). Users can create their functions with different models as well.
`dates`	vector of class "Date" with all days of observations (yyyy-mm-dd).
`stmethod`	standardization method. 'quant' or 'ratio', see details.
`thres`	numeric. Maximum radius (in km) where neighboring stations will be searched. NA value uses the whole spatial domain.
`neibs`	integer. Number of nearest neighbors to use.
`coords`	vector of two character elements. Names of the fields in "sts" containing longitude and latitude.
`crs`	character. Coordinates system in EPSG format (e.g.: "EPSG:4326").
`coords_as_preds`	logical. If TRUE (default), "coords" are also taken as predictors.
`window`	odd integer. Length of data considered for standardization
`ncpu`	number of processor cores used to parallel computing.

Details

After the gap filling, "stmethod" allows for an standardization of the predictions based on the observations. It only works for daily data. For other timescales (monthly, annual) use "stmethod=NULL". The "window" parameter is a daily-moving centered window from which data is collected for each year (i.e. a 15-day window on 16th January will take all predictions from 1st to 30th January of all years to standardize them with their corresponding observations. Only standardized prediction of 16th January is returned. Process is repeated for all days).

Examples

```
## Not run:
set.seed(123)
prec <- round(matrix(rnorm(30*50, mean = 1.2, sd = 6), 30, 50), 1)
prec[prec<0] <- 0
prec <- apply(prec, 2, FUN = function(x){x[sample(length(x),5)] <- NA; x})
colnames(prec) <- paste0('sts_',1:50)
sts <- data.frame(ID = paste0('sts_',1:50), lon = rnorm(50,0,1),
                  lat = rnorm(50,40,1), dcoast = rnorm(50,200,50))
filled <- gapFilling(prec, sts,
                    dates = seq.Date(as.Date('2023-04-01'),
                    as.Date('2023-04-30'),by='day'),
                    stmethod = "ratio", thres = NA, coords = c('lon','lat'),
                    coords_as_preds = TRUE, crs = 'EPSG:4326', neibs = 10,
                    window = 11, ncpu = 2)
str(filled)
summary(filled)

## End(Not run)
```

`gridPcp` *Gridded dataset creation*

Description

This function creates a gridded precipitation dataset from a station-based dataset.

Usage

```
gridPcp(
  prec,
  grid,
  dyncovars = NULL,
  sts,
  model_fun,
  dates,
  ncpu,
  thres,
  neibs,
  coords,
  crs,
  coords_as_preds,
  dir_name = NA
)
```

Arguments

`prec`	matrix or data.frame containing the original (cleaned) precipitation data. Each column represents one station. The names of columns must coincide with the names of the stations.
`grid`	SpatRaster. Collection of rasters representing each one of the predictors.
`dyncovars`	SpatRasterDataset. Collection of variables acting as dynamic predictors (changing each day). Each dataset inside the sds object (corresponding to each variable) must have the same rasters as number of days to grid. See Details.
`sts`	matrix or data.frame. A column "ID" (unique ID of stations) is required. The rest of the columns (all of them) will act as predictors of the model.
`model_fun`	function. A function that integrates the statistical hybrid model (classification and regression). The default is learner_glm, which is the original model. Other models are also available (learner_rf and learner_xgboost). Users can create their functions with different models as well.

`dates`	vector of class "Date" with all days of observations (yyyy-mm-dd).
`ncpu`	number of processor cores used to parallel computing.
`thres`	numeric. Maximum radius (in km) where neighboring stations will be searched. NA value uses the whole spatial domain.
`neibs`	integer. Number of nearest neighbors to use.
`coords`	vector of two character elements. Names of the fields in "sts" containing longitude and latitude.
`crs`	character. Coordinates system in EPSG format (e.g.: "EPSG:4326").
`coords_as_preds`	logical. If TRUE (default), "coords" are also taken as predictors.
`dir_name`	character. Name of the of the folder in which the data will be saved. Default NA uses the original names.

Details

All the rasters provided in "grid" and "dyncovars" must have the same spatial characteristics (resolution, crs, extent, etc.). The function estimates precipitation based on the nearest observations ("sts" and "prec") using as covariates the predictors contained in "grid" and "dyncovars". Predictors of "grid" are used for all days while those on "dyncovars" are selected depending on the day. For instance, to model the first day the algorithm considers all rasters in "grid" and only those corresponding to the first day of each variable.

Examples

```
## Not run:
# fixed covariates (elevation, latitude, longitude)
alt <- terra::rast(volcano, crs = 'EPSG:4326')
terra::ext(alt) <- c(-1,3,38,42)
lon <- terra::rast(cbind(terra::crds(alt),terra::crds(alt)[,1]),type='xyz',crs='EPSG:4326')
lat <- terra::rast(cbind(terra::crds(alt),terra::crds(alt)[,2]),type='xyz',crs='EPSG:4326')
dcoast <- terra::costDist(alt,target=min(terra::values(alt)))/1000
grid <- c(alt, lon, lat, dcoast)
names(grid) <- c('alt', 'lon', 'lat', 'dcoast')

# Dynamic covariates (Variable 1, Variable 2, Variable 3)
foo <- alt
terra::values(foo) <- runif(length(terra::values(alt)))
dyncovars1 <- rep(foo, 7)
names(dyncovars1) <- paste('dynvar1.day',1:terra::nlyr(dyncovars1),
                      sep = "_") # not use blank space!
dyncovars2 <- dyncovars1*0.0234
names(dyncovars2) <- paste('dynvar2.day',1:terra::nlyr(dyncovars2),
                      sep = "_") # not use blank space!
dyncovars3 <- dyncovars1*10502
names(dyncovars3) <- paste('dynvar3.day',1:terra::nlyr(dyncovars3),
                      sep = "_") # not use blank space!
dyncovars <- terra::sds(dyncovars1, dyncovars2, dyncovars3)

# precipitation and stations generation
```

```
set.seed(123)
prec <- round(matrix(rnorm(7*25, mean = 1.2, sd = 4), 7, 25), 1)+1
prec[prec<0] <- 0
colnames(prec) <- paste0('sts_',1:25)
sts <- data.frame(ID = paste0('sts_',1:25), as.data.frame(terra::spatSample(grid, 25)))


gridPcp(prec = prec,
        grid = grid,
        sts = sts,
        model_fun = learner_glm,
        dates = seq.Date(as.Date('2023-04-01'),as.Date('2023-04-07'),by='day'),
        ncpu = 4,
        thres = NA,
        neibs = 15,
        coords = c('lon','lat'),
        crs = 'EPSG:4326',
        coords_as_preds = TRUE)

r <- terra::rast(c('./pred/20230401.tif','./err/20230401.tif'))
terra::plot(r)

## End(Not run)
```

hmg_Ts *Homogenization of daily precipitation time series*

Description

The function applies the homogenization procedure (detection and adjustment) for daily precipitation based on Huerta et al. (2024).

Usage

```
hmg_Ts(
  prec,
  sts,
  neibs_max = 8,
  neibs_min = 3,
  thres = 1e+06,
  cor_neibs = 0.5,
  cleaning = FALSE,
  perc_break = 7,
  wet_day = 0,
  window_c = 15,
  apply_qc = 1,
  mm_apply_qc = 0,
  ncpu = 2
)
```

Arguments

`prec`	xts matrix containing the raw (quality-controlled and gap-filled) precipitation data. Each column represents one station. The names of columns must coincide with the names of the stations.
`sts`	data.frame. A column "ID" (unique ID of stations), "LON" (decimal degree), and "LAT" (decimal degree) are required.
`neibs_max`	integer. Number of maximum nearest neighbors to use. NA or Null value uses the whole number of stations
`neibs_min`	integer. Number of minimum nearest neighbors to use. This value represent the option to use the absolute or relative approach on the homogenization.
`thres`	numeric. Maximum radius (in m) where neighboring stations will be searched. NA or Null value uses the whole spatial domain.
`cor_neibs`	numeric. Minimum value of temporal correlation to define nearest neighbors.
`cleaning`	logical. Set to FALSE as default. TRUE if the time series should be cleaned (trends and autocorrelation removed) before the detection test, as in Lund et al. (2023).
`perc_break`	numeric. Value that define the percentage of time series that are statistically significant in the detection test in order to define a break point (year).
`wet_day`	numeric. Value that define if the adjustment should be only performed on wet day (> 0 mm). Negative value mean that the adjustment will be also applied on zeros.
`window_c`	integer. Window size of the application of the quantile matching adjustment.
`apply_qc`	numeric. Maximum threshold (cubic difference) in which the adjusted data be considered corrected. Set to 1 as default. If the difference of the root cubic between the adjusted and raw data is above 1, the adjusted value will be corrected to be not above (or below) 1.
`mm_apply_qc`	numeric. Precipitation threshold in which apply_qc would be applied. Set to 0 as default, meaning that only the adjustment will be on wet days (> 0 mm)
`ncpu`	number of processor cores used to parallel computing.

Details

The homogenization procedure uses an automatic algorithm for both detection and adjustment without metadata information. In addition, relative and absolute approaches are combined for situations in which relative homogenization can not be performed. The absolute test, which has a lower power of detection than the relative tests, is thus intended as a backup test for when a relative test is hardly possible. To ensure high confidence in breakpoint detection, a combination of different statistical tests and intercomparison of their results was used. Five univariate breakpoint tests were applied: Student's, Mann-Whitney, Buishand-R, Pettit, and the Standard Normal Homogeneity Test. Depending on the availability of nearby stations for a target time series: relative and absolute. For the relative approach, the algorithm searches for up to neibs_max well-correlated (> cor_neibs) nearby stations within a three radius. Later, the five tests are applied to difference series (target minus nearby) created with three different temporal aggregations. Finally, the breakpoint is set to a certain year if it is found in at least perc_break (%) of the number of difference time series that are significant (p-value < 0.05), using a tolerance of +/- 1 year. The absolute approach is used if the algorithm detects fewer than neibs_min nearby stations or none at all. In the adjustment, it was adapted to the quantile-matching technique outlined in Squintu et al. (2018). It should be mentioned that this algorithm was created for temperature data; therefore, we made some changes to be used for precipitation. Dry values (wet_day parameter) cannot be corrected, and wet values were transformed twice (square root and log) before the algorithm execution to force a normal distribution. Based on this consideration, the correction was applied in two ways: relative and absolute. For the relative approach, the adjustment factor was computed using the target and nearby time series of the detection stage. It is assumed that the data after the break is correct; thus, the correction is backward. For the absolute case, the adjustment factor was computed using the target time series. This can be seen as an application of quantile mapping as there are no nearby stations. Adjustment of daily precipitation can influence the extreme tails. Therefore, adjusted values can be set to not exceed a limit (apply_qc - difference of the root cubic) with the raw data. The apply_qc can be applied to all precipitation values or above a specific threshold: mm_apply_qc. This procedure still keeps the extreme adjustment while preventing the creation of extremely excessive values. The output of the function is a list that contains: the detection results and the adjusted time series for each station.

References

Huerta, A., Serrano-Notivoli, R., & Brönnimann, S. (2024). SC-PREC4SA: A serially complete daily precipitation dataset for South America. https://doi.org/10.31223/X57D8R

Lund, R. B., Beaulieu, C., Killick, R., Lu, Q., & Shi, X. (2023). Good practices and common pitfalls in climate time series changepoint techniques: A review. Journal of Climate, 36(23), 8041-8057.

Squintu, A. A., van der Schrier, G., Brugnara, Y., & Klein Tank, A. (2018). Homogenization of daily ECA&D temperature series. International journal of climatology, 39(3), 1243-1261.

`learner_glm` *Function of the generalized linear model (glm)*

Description

The function of the generalized linear model (glm) is used to predict wet/dry (classification) days and the amount of rain on wet days (regression). This function employs the hybrid model of Serrano-Notivoli et al. (2017).

Usage

```
learner_glm(ref, can, covars)
```

Arguments

`ref`	matrix or data.frame containing the covariable data and precipitation value for each point location. This data is used to train (build) the model
`can`	matrix or data.frame containing the variable data and precipitation value for one single point location. This data is used to execute the build model and get the predictions
`covars`	vector of character elements containing the covariables names used in the model.

Details

The function should not be used directly, as its main purpose is to be passed as an argument in the core functions (qcPrec, gapFilling and gridPcp) of the package. The core functions can handle whatever condition that avoids the execution of this function. Therefore, there is no need to test different situations on the data (empty values, among others). However, users who want to build their functions should test first before being passed as an argument in the core functions (you can check the other models (learner) functions available in the package). This function encapsulates two types of models: classification and regression, and its output is a single numeric vector of three elements: probability of wet day, amount of wet day, and uncertainty of the amount of wet day.

References

Serrano-Notivoli, R., Beguería, S., Saz, M. Á., Longares, L. A., & de Luis, M. (2017). SPREAD: a high-resolution daily gridded precipitation dataset for Spain–an extreme events frequency and intensity overview. Earth System Science Data, 9(2), 721-738.

Examples

```
## Not run:
set.seed(123)
# creating random data (three predictors)
lon = rnorm(50,0,1)
lat = rnorm(50,40,1)
dcoast = rnorm(50,200,50)
prec = rnorm(1*50, mean = 1.2, sd = 6)
prec[prec<0] <- 0

# precipitation column should be call as "val"
data_full <- data.frame(lon = lon, lat = lat, dcoast = dcoast, val = prec)

# parameters for learner_glm
ref = data_full[-1, ]
can = data_full[1, ]
covars = c("lon", "lat", "dcoast")

learner_glm(ref = ref, can = can, covars = covars)

# case when "prec" is full wet
prec = rnorm(1*50, mean = 1.2, sd = 6)
prec[prec<0] <- 1

data_full <- data.frame(lon = lon, lat = lat, dcoast = dcoast, val = prec)

# parameters for learner_glm
ref = data_full[-1, ]
can = data_full[1, ]
covars = c("lon", "lat", "dcoast")

learner_glm(ref = ref, can = can, covars = covars)

## End(Not run)
```

learner_nn *Function of the neural network model (nn)*

Description

The function of the neural network model (nn) is used to predict wet/dry (classification) days and the amount of rain on wet days (regression). This function employs the hybrid model of Serrano-Notivoli et al. (2017).

Usage

```
learner_nn(ref, can, covars)
```

Arguments

`ref`	matrix or data.frame containing the covariable data and precipitation value for each point location. This data is used to train (build) the model
`can`	matrix or data.frame containing the variable data and precipitation value for one single point location. This data is used to execute the build model and get the predictions
`covars`	vector of character elements containing the covariables names used in the model.

Details

The function should not be used directly, as its main purpose is to be passed as an argument in the core functions (qcPrec, gapFilling and gridPcp) of the package. The core functions can handle whatever condition that avoids the execution of this function. Therefore, there is no need to test different situations on the data (empty values, among others). However, users who want to build their functions should test first before being passed as an argument in the core functions (you can check the other models (learner) functions available in the package). This function encapsulates two types of models: classification and regression, and its output is a single numeric vector of three elements: probability of wet day, amount of wet day, and uncertainty of the amount of wet day. To use this function, the neuralnet package should have been installed previously

References

Huerta, A., Serrano-Notivoli, R., & Brönnimann, S. (2024). SC-PREC4SA: A serially complete daily precipitation dataset for South America. https://doi.org/10.31223/X57D8R

Examples

```
## Not run:
set.seed(123)
# creating random data (three predictors)
lon = rnorm(50,0,1)
lat = rnorm(50,40,1)
dcoast = rnorm(50,200,50)
prec = rnorm(1*50, mean = 1.2, sd = 6)
prec[prec<0] <- 0

# precipitation column should be call as "val"
data_full <- data.frame(lon = lon, lat = lat, dcoast = dcoast, val = prec)

# parameters for learner_nn
ref = data_full[-1, ]
can = data_full[1, ]
covars = c("lon", "lat", "dcoast")

learner_nn(ref = ref, can = can, covars = covars)

# case when "prec" is full wet
prec = rnorm(1*50, mean = 1.2, sd = 6)
prec[prec<0] <- 1

data_full <- data.frame(lon = lon, lat = lat, dcoast = dcoast, val = prec)

# parameters for learner_nn
ref = data_full[-1, ]
can = data_full[1, ]
covars = c("lon", "lat", "dcoast")

learner_nn(ref = ref, can = can, covars = covars)

## End(Not run)
```

`learner_rf` *Function of the random forest model (rf)*

Description

The function of the random forest model (rf) is used to predict wet/dry (classification) days and the amount of rain on wet days (regression). This function employs the hybrid model of Huerta et al. (2024).

Usage

```
learner_rf(ref, can, covars)
```

Arguments

`ref`	matrix or data.frame containing the covariable data and precipitation value for each point location. This data is used to train (build) the model
`can`	matrix or data.frame containing the variable data and precipitation value for one single point location. This data is used to execute the build model and get the predictions
`covars`	vector of character elements containing the covariables names used in the model.

Details

The function should not be used directly, as its main purpose is to be passed as an argument in the core functions (qcPrec, gapFilling and gridPcp) of the package. The core functions can handle whatever condition that avoids the execution of this function. Therefore, there is no need to test different situations on the data (empty values, among others). However, users who want to build their functions should test first before being passed as an argument in the core functions (you can check the other models (learner) functions available in the package). This function encapsulates two types of models: classification and regression, and its output is a single numeric vector of three elements: probability of wet day, amount of wet day, and uncertainty of the amount of wet day. To use this function, the randomForest package should have been installed previously

References

Huerta, A., Serrano-Notivoli, R., & Brönnimann, S. (2024). SC-PREC4SA: A serially complete daily precipitation dataset for South America. https://doi.org/10.31223/X57D8R

Examples

```
## Not run:
set.seed(123)
# creating random data (three predictors)
lon = rnorm(50,0,1)
lat = rnorm(50,40,1)
dcoast = rnorm(50,200,50)
prec = rnorm(1*50, mean = 1.2, sd = 6)
prec[prec<0] <- 0

# precipitation column should be call as "val"
data_full <- data.frame(lon = lon, lat = lat, dcoast = dcoast, val = prec)

# parameters for learner_rf
ref = data_full[-1, ]
can = data_full[1, ]
covars = c("lon", "lat", "dcoast")

learner_rf(ref = ref, can = can, covars = covars)

# case when "prec" is full wet
prec = rnorm(1*50, mean = 1.2, sd = 6)
prec[prec<0] <- 1
```

```
data_full <- data.frame(lon = lon, lat = lat, dcoast = dcoast, val = prec)

# parameters for learner_rf
ref = data_full[-1, ]
can = data_full[1, ]
covars = c("lon", "lat", "dcoast")

learner_rf(ref = ref, can = can, covars = covars)

## End(Not run)
```

`learner_svm` *Function of the support vector machine model (svm)*

Description

The function of the support vector machine model (svm) is used to predict wet/dry (classification) days and the amount of rain on wet days (regression). This function employs the hybrid model of Serrano-Notivoli et al. (2017).

Usage

```
learner_svm(ref, can, covars)
```

Arguments

`ref`	matrix or data.frame containing the covariable data and precipitation value for each point location. This data is used to train (build) the model
`can`	matrix or data.frame containing the variable data and precipitation value for one single point location. This data is used to execute the build model and get the predictions
`covars`	vector of character elements containing the covariables names used in the model.

Details

The function should not be used directly, as its main purpose is to be passed as an argument in the core functions (qcPrec, gapFilling and gridPcp) of the package. The core functions can handle whatever condition that avoids the execution of this function. Therefore, there is no need to test different situations on the data (empty values, among others). However, users who want to build their functions should test first before being passed as an argument in the core functions (you can check the other models (learner) functions available in the package). This function encapsulates two types of models: classification and regression, and its output is a single numeric vector of three elements: probability of wet day, amount of wet day, and uncertainty of the amount of wet day. To use this function, the e1071 package should have been installed previously

References

Huerta, A., Serrano-Notivoli, R., & Brönnimann, S. (2024). SC-PREC4SA: A serially complete daily precipitation dataset for South America. https://doi.org/10.31223/X57D8R

Examples

```
## Not run:
set.seed(123)
# creating random data (three predictors)
lon = rnorm(50,0,1)
lat = rnorm(50,40,1)
dcoast = rnorm(50,200,50)
prec = rnorm(1*50, mean = 1.2, sd = 6)
prec[prec<0] <- 0

# precipitation column should be call as "val"
data_full <- data.frame(lon = lon, lat = lat, dcoast = dcoast, val = prec)

# parameters for learner_svm
ref = data_full[-1, ]
can = data_full[1, ]
covars = c("lon", "lat", "dcoast")

learner_svm(ref = ref, can = can, covars = covars)

# case when "prec" is full wet
prec = rnorm(1*50, mean = 1.2, sd = 6)
prec[prec<0] <- 1

data_full <- data.frame(lon = lon, lat = lat, dcoast = dcoast, val = prec)

# parameters for learner_svm
ref = data_full[-1, ]
can = data_full[1, ]
covars = c("lon", "lat", "dcoast")

learner_svm(ref = ref, can = can, covars = covars)

## End(Not run)
```

`learner_xgboost` *Function of the eXtreme Gradient Boosting model (xgboost)*

Description

The function of the eXtreme Gradient Boosting model (xgboost) is used to predict wet/dry (classification) days and the amount of rain on wet days (regression). This function employs the hybrid model of Huerta et al. (2024).

Usage

```
learner_xgboost(ref, can, covars)
```

Arguments

`ref`	matrix or data.frame containing the covariable data and precipitation value for each point location. This data is used to train (build) the model
`can`	matrix or data.frame containing the variable data and precipitation value for one single point location. This data is used to execute the build model and get the predictions
`covars`	vector of character elements containing the covariables names used in the model.

Details

The function should not be used directly, as its main purpose is to be passed as an argument in the core functions (qcPrec, gapFilling and gridPcp) of the package. The core functions can handle whatever condition that avoids the execution of this function. Therefore, there is no need to test different situations on the data (empty values, among others). However, users who want to build their functions should test first before being passed as an argument in the core functions (you can check the other models (learner) functions available in the package). This function encapsulates two types of models: classification and regression, and its output is a single numeric vector of three elements: probability of wet day, amount of wet day, and uncertainty of the amount of wet day. To use this function, the xgboost package should have been installed previously

References

Huerta, A., Serrano-Notivoli, R., & Brönnimann, S. (2024). SC-PREC4SA: A serially complete daily precipitation dataset for South America. https://doi.org/10.31223/X57D8R

Examples

```
## Not run:
set.seed(123)
# creating random data (three predictors)
lon = rnorm(50,0,1)
lat = rnorm(50,40,1)
dcoast = rnorm(50,200,50)
prec = rnorm(1*50, mean = 1.2, sd = 6)
prec[prec<0] <- 0

# precipitation column should be call as "val"
data_full <- data.frame(lon = lon, lat = lat, dcoast = dcoast, val = prec)

# parameters for learner_xgboost
ref = data_full[-1, ]
can = data_full[1, ]
covars = c("lon", "lat", "dcoast")

learner_xgboost(ref = ref, can = can, covars = covars)
```

```
# case when "prec" is full wet
prec = rnorm(1*50, mean = 1.2, sd = 6)
prec[prec<0] <- 1

data_full <- data.frame(lon = lon, lat = lat, dcoast = dcoast, val = prec)

# parameters for learner_xgboost
ref = data_full[-1, ]
can = data_full[1, ]
covars = c("lon", "lat", "dcoast")

learner_xgboost(ref = ref, can = can, covars = covars)

## End(Not run)
```

qcPrec *Quality Control of daily precipitation observations*

Description

This function apply several threshold-based criteria to filter original observations of daily precipitation.

Usage

```
qcPrec(
  prec,
  sts,
  model_fun = learner_glm,
  crs,
  coords,
  coords_as_preds = TRUE,
  neibs = 10,
  thres = NA,
  qc = "all",
  qc3 = 10,
  qc4 = c(0.99, 5),
  qc5 = c(0.01, 0.1, 5),
  ncpu = 1
)
```

Arguments

`prec`	matrix containing the original precipitation data. Each column represents one station. The names of columns have to be names of the stations.
`sts`	data.frame. A column "ID" (unique ID of stations) is required. The rest of the columns (all of them) will act as predictors of the model.

`model_fun`	function. A function that integrates the statistical hybrid model (classification and regression). The default is learner_glm, which is the original model. Other models are also available (learner_rf and learner_xgboost). Users can create their functions with different models as well.
`crs`	character. Coordinates system in EPSG format (e.g.: "EPSG:4326").
`coords`	vector of two character elements. Names of the fields in "sts" containing longitude and latitude.
`coords_as_preds`	logical. If TRUE (default), "coords" are also taken as predictors.
`neibs`	integer. Number of nearest neighbors to use.
`thres`	numeric. Maximum radius (in km) where neighboring stations will be searched. NA value uses the whole spatial domain.
`qc`	vector of strings with the QC criteria to apply. Default is "all". See details.
`qc3`	numeric. Indicates the threshold (number of times higher or lower) from which a observation, in comparison with its estimate, should be deleted. Default is 10.
`qc4`	numeric vector of length 2. Thresholds of wet probability (0 to 1) and magnitude (in the units of input precipitation data) from which a observation of value zero, in comparison with its estimate, should be deleted. Default is c(0.99, 5).
`qc5`	numeric vector of length 2. Thresholds of dry probability (0 to 1) and magnitude (in the units of input precipitation data) from which a observation higher than a specific value (also in the original units), in comparison with its estimate, should be deleted. Default is c(0.01, 0.1, 5).
`ncpu`	number of processor cores used to parallel computing.

Details

Parameter "sts" must have an "ID" field containing unique identifiers of the stations.

"qc" can be "all" (all criteria are applied) or a vector of strings (e.g.: c("1","2","4")) indicating the QC criteria to apply to observations: "1" (suspect value): obs==0 & all(neibs>0); "2" (suspect zero): obs>0 & all(neibs==0); "3" (suspect outlier): obs is "qc3" times higher or lower than the estimate; "4" (suspect wet): obs==0 & wet probability > "qc4[1]" & estimate > "qc4[2]"; "5" (suspect dry): obs>"qc5[3]" & dry probability < "qc5[1]" & estimate < "qc5[2]"

Examples

```
## Not run:
set.seed(123)
prec <- round(matrix(rnorm(30*50, mean = 1.2, sd = 6), 30, 50), 1)
prec[prec<0] <- 0
colnames(prec) <- paste0('sts_',1:50)
sts <- data.frame(ID = paste0('sts_',1:50), lon = rnorm(50,0,1),
                  lat = rnorm(50,40,1), dcoast = rnorm(50,200,50))
qcdata <- qcPrec(prec, sts, crs = 'EPSG:4326', coords = c('lon','lat'),
                 coords_as_preds = TRUE, neibs = 10, thres = NA,
                 qc = 'all', qc3 = 10, qc4 = c(0.99, 5), qc5 = c(0.01, 0.1, 5),
                 ncpu=2)
str(qcdata)
```

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

Index