Package: agroclimatico (via r-universe)

December 5, 2024

Title Índices y Estadísticos Climáticos e Hidrológicos

Version 1.1.0

Description Conjunto de funciones para calcular índices y estadísticos climáticos hidrológicos a partir de datos tidy. Incluye una función para graficar resultados georeferenciados y e información cartográfica.

License GPL-3 + file LICENSE

Encoding UTF-8

Language es

LazyData true

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

URL https://github.com/ropensci/agroclimatico, https://docs.ropensci.org/agroclimatico/

BugReports https://github.com/ropensci/agroclimatico/issues

Depends R (>= 2.10)

Imports readr, data.table, scales, lmomco, sf, automap, png, grid, ggnewscale, tidyr, ggplot2, rappdirs, magrittr, kableExtra, Rcpp (>= 0.12.0), cli, SPEI (>= 1.8.1)

Suggests testthat, dplyr, vdiffr, lubridate, covr, knitr, rmarkdown

VignetteBuilder knitr

LinkingTo Rcpp

Config/pak/sysreqs libfontconfig1-dev libfreetype6-dev libgdal-dev gdal-bin libgeos-dev make libicu-dev libpng-dev libxml2-dev libssl-dev pari-gp libproj-dev libsqlite3-dev libudunits2-dev libx11-dev

Repository https://ropensci.r-universe.dev

RemoteUrl https://github.com/ropensci/agroclimatico

RemoteRef master

RemoteSha 246e6b62fd2941cf3ef87736d95392fc497dac09

Contents

`agromet_informe` *Formato de salida para Informes*

Description

Los formatos utilizan como base `rmarkdown::pdf_document()` y una plantilla específica de LaTex.

Usage

```
agromet_informe(..., latex_engine = "xelatex")
```

Arguments

`...`	cualquier argumento que requiera `rmarkdown::pdf_document()`.
`latex_engine`	Caracter con el compilador de latex a usar.

Value

documento compilado.

Examples

```
## Not run:
agromet_informe("Informe.Rmd")

## End(Not run)
```

`completar_serie` *Completa una Serie de Datos*

Description

La función permite completar una serie de datos temporales definiendo alguna resolución disponible. Es compatible con datos agrupados por `dplyr::group_by()`.

Usage

```
completar_serie(datos, fecha, resolucion, rango = range(fecha))
```

Arguments

`datos`	tabla (data.frame, data.table, tibble) a completar.
`fecha`	variable tipo fecha (Date, IDate, POSIXct, etc).
`resolucion`	texto que define resolución de salida de los datos. Puede ser cualquier valor aceptado por el argumento by de la función `seq.Date()` o su traducción al español. Es decir, "día" (o "dia"), "semana", "mes", "trimestre" o "año", así como los plurales.
`rango`	un vector cuyo rango define el período a completar. Es útil si se quiere que múltiples grupos de datos tengan el mismo rango de fechas. Es posible definir un rango por fuera del rango original de los datos para homogeneizar series temporales.

Value

Devuelve un data.frame con las mismas variables de origen. La variable asociada a las fechas ahora se encuentra completa para la resolución indicada y el resto de las variables se completan con NA.

Examples

```
# Datos de prueba completos
datos <- data.frame(fechas = seq(as.Date("2013-01-01"), as.Date("2015-12-01"),
                                 by = "1 month"),
                    pp = 1)

set.seed(42)
datos_perdidos <- sample(nrow(datos), nrow(datos)/10)
datos$pp[datos_perdidos] <- NA
```

```
datos_incompletos <- na.omit(datos) #Serie de datos a completar

completar_serie(datos_incompletos, fechas, resolucion = "1 mes")
```

`datos_nh` *Observaciones diarias de estaciones meteorológicas.*

Description

Los datos provienen de 2 estaciones del INTA. Ver `metadatos_nh()` para más información y metadatos de las estaciones.

Usage

```
NH0358

NH0114
```

Format

NH0358 es un data.table con 25515 filas y 25 columnas

NH0114 es un data.table con 24041 filas y 25 columnas

See Also

`metadatos_nh()` devuelve los metadatos de las estaciones meteorológicas.

`datos_nh_mensual` *Observaciones Meteorológicas*

Description

Datos mensuales de temperatura y precipitación en estaciones meteorológicas en Argentina.

Usage

```
datos_nh_mensual
```

Format

Un data.table con 381 filas y 10 columnas

mes Fecha en formato %Y-%m-%d

precipitacion_mensual Suma de la precipitación durante el mes en mm.

temperatura_media_mensual Promedio de la temperatura en el mes en grados celsius.

codigo_nh Código de identificación de la estación

estacion Nombre de la estación.

provincia Provincia donde se ubica la estación.

organismo Organismo a cargo de la estación.

lat Latitud.

lon Longitud.

altura Altura sobre el nivel del mar donde se ubica la estación.

`decil` *Transformaciones estadísticas de variables*

Description

Las funciones `decil()` y `anomalia_porcentual()` devuelven estos estadísticos para alguna variable dado un periodo de referencia especificado.

Usage

```
decil(variable, referencia = rep(TRUE, length(variable)))

anomalia_porcentual(
  variable,
  referencia = rep(TRUE, length(variable)),
  na.rm = FALSE
)
```

Arguments

`variable`	vector de observaciones de la variable de interés.
`referencia`	serie de observaciones para usar de referencia en el ajuste a la distribución teórica. Puede ser: • vector lógico que se usará para filtrar los datos de entrada. • vector numérico de observaciones. • la serie completa (opción por defecto).
`na.rm`	lógico. Define si se utilizan o no valores faltantes en el cálculo de la anomalía porcentual.

Details

Para recuperar el valor de la variable asociado a determinado decil se puede utilizar la función `stats::quantile()`.

Value

Para el cálculo de los deciles, devuelve un vector numérico con el decil asociado a cada valor de la variable. En este caso la columna `deciles` es de tipo doble ya que devuelve el valor exacto del decil sin redondeos. Para el cálculo de la anomalía porcentual también devuelve un vector numérico. Las funciones son compatibles con `dplyr::group_by()` y `dplyr::mutate()`.

Examples

```
library(dplyr)
data(NH0358)

# Deciles de precipitación usando como referencia la serie completa
NH0358 %>%
  mutate(deciles = decil(precip)) %>%
  slice_head(n = 10)

# Deciles mensuales
precip_mensual <- NH0358 %>%
  group_by(fecha = lubridate::floor_date(fecha, "month")) %>%
  summarise(precip = sum(precip, na.rm = TRUE))

precip_mensual %>%
  mutate(deciles = decil(precip))

# Deciles definiendo un periodo de referencia
precip_mensual %>%
  mutate(deciles = decil(precip,
                         referencia = lubridate::year(fecha) <= 1958))

# Anomalia porcentual usando como referencia la serie completa
NH0358 %>%
  mutate(anomalia = anomalia_porcentual(precip)) %>%
  slice_head(n = 10)

#  Anomalia porcentual definiendo un periodo de referencia
precip_mensual %>%
  mutate(deciles = anomalia_porcentual(precip,
                         referencia = lubridate::year(fecha) <= 1958))
```

`dias_promedio` *Días Promedio*

Description

Calcula el primer y último día del año promedio a partir de una serie de fechas.

Usage

```
dias_promedio(fechas)
```

Arguments

`fechas`	vector de fechas

Details

Esta función solo requiere un vector de fechas para calcular el primer y último día del año en promedio. Si este vector incluye todos los días de muchos años el resultado será el 1° de enero y el 31 de diciembre. Pero si solo se utilizan las fechas que cumplen con una determinada condición, por ejemplo aquellos días donde la temperatura mínima fue menor o igual a 0°C, entonces devuelve el primer y último día de ocurrencia en promedio para este evento.

La función se puede usar tanto con la sintaxis de base como con dplyr (ver ejemplos). En el caso de dplyr es necesario usar la función `dplyr::reframe()` ya que `dias_promedio()` devuelve un data.frame. Es posible hacer cálculos agrupando datos con `dplyr::group_by()`.

Value

La función devuelve un data.frame con 4 variables fijas y variables extras en el caso de hacer el cálculo para distintos grupos:

- `variable` (caracter) primer_dia o ultimo_dia según corresponda
- `dia` (numérico) día del mes
- `mes` (numérico) mes de ocurrencia
- `dia_juliano` (numérico) día del año

Examples

```
data(NH0358)

# Usando la serie completa
dias_promedio(NH0358$fecha)

# Filtrando los datos para un determinado evento
library(dplyr)
NH0358 %>%
  filter(t_min <= 0) %>%
  reframe(dias_promedio(fecha))

# Por grupos, si tenemos por ejemplo más de una estación
data(NH0114)

rbind(NH0358, NH0114) %>%
```

```
  filter(t_min <= 0) %>%
  group_by(codigo_nh) %>%
  reframe(dias_promedio(fecha))
```

`escala_temp_min` *Escalas de colores usadas por el INTA*

Description

Escalas de colores típicas usadas por INTA para distintas variables.

Usage

```
escala_temp_min

escala_temp_max

escala_pp_mensual

escala_pp_diaria
```

Format

Objeto tipo lista con 3 elementos.

Value

Lista en el mismo formato que devuelve `leer_surfer()` con elementos:

- `niveles` (numérico), el nivel a que corresopnde cada color.
- `colores` (caracter), la representación hexadecimal del color de cada break.
- `paleta` (función), una función que toma un entero `n` y devuelve un vector de caracter con `n` colores interpolados a partir de los colores de la escala.

Examples

```
library(ggplot2)
library(dplyr)

pp_enero <- datos_nh_mensual |>
  filter(mes == unique(mes)[1])

# En el contexto de la función mapear():
mapear(pp_enero, precipitacion_mensual, lon, lat,
escala = escala_pp_mensual, cordillera = TRUE)

# Con ggplot2
# Los contornos llenos requieren que los datos estén en una grilla
```

```
# regular, necesitamos hacer una interpolación con kriging.
with(pp_enero, agroclimatico:::kringe(precipitacion_mensual, lon, lat)) |>
ggplot(aes(lon, lat)) +
 geom_contour(aes(z = var1.pred)) +
 geom_contour_filled(aes(z = var1.pred)) +
 scale_fill_inta(escala = escala_pp_mensual)
```

`ith` *Índice de Temperatura y Humedad*

Description

Calcula el índice de temperatura y humedad (ITH)

Usage

```
ith(temperatura, hr)
```

Arguments

`temperatura`	vector numérico con valores (o valor) de temperatura en grados centígrados.
`hr`	vector númerico (o valor) de la misma longitud que temperatura con la humedad relativa en porcentaje.

Value

Devuelve un valor o vector de valores con el ITH. Este valor es utilizado como una medida de la intensidad de las condiciones de estrés por calor a la que se encuentra expuesto el animal. Para bovinos se categoriza como:

- **Normal** si ITH < 75
- **Alerta** para ITH entre 75 y 78
- **Peligro** para ITH entre 79 y 83
- **Emergencia** para ITH >= 84

References

- Armendano, J. I. ¿Cuándo se generan condiciones de estrés por calor en bovinos para carne? link
- Armstrong, DV. 1994. Heat stress interaction with shade and cooling. J. Diary Sci. 77:2004-2050

Examples

```
ith(temperatura = 23, hr = 65)

data(NH0358)

# En el contexto de mutate
library(dplyr)
NH0358 %>%
  filter(!is.na(hr)) %>%
  mutate(t_media = (t_max + t_min)/2) %>%
  mutate(ith = ith(t_media, hr)) %>%
  slice_head(n = 10)
```

`kable_inta` *Formatea tablas con el estilo de INTA*

Description

Llama a `kableExtra::kbl()` con valores por defecto apropiados acordes al estilo utilizado por INTA.

Usage

```
kable_inta(x, ...)
```

Arguments

`x`	Una tabla.
`...`	Otros argumentos que se pasan a `kableExtra::kbl()`

Value

tabla, objeto kbl.

Examples

```
library(dplyr)
library(kableExtra)
metadatos <- metadatos_nh()

metadatos %>%
 head() %>%
 select(codigo_nh, estacion) %>%
 kable_inta(caption = "Ejemplo",
           col.names = c("Código", "Estación")) %>%
 kable_styling(latex_options = "scale_down")
```

`leer_nh` *Lectura de Archivos con Formato NH*

Description

Lee uno o más archivos siempre que mantengan el formato NH de ancho fijo.

Usage

```
leer_nh(archivos)
```

Arguments

`archivos`	Caracter o vector de caracteres con nombre y ubicación de los archivos a leer.

Details

La función está preparada para leer datos diarios con el formato NH que incluye 25 variables:

- `codigo` (caracter)
- `codigo_nh` (caracter), variable llave para acceder a los metadatos de estaciones
- `fecha` (fecha)
- `t_max` (numérico), temperatura máxima en grados centígrados
- `t_min` (numérico), temperatura mínima en grados centígrados
- `precip` (numérico), precipitación acumulada en milímetros
- `lluvia_datos` (numérico), ocurrencia de precipitación 1 indica lluvia
- `lluvia` (numérico), ocurrencia de lluvia
- `llovizna` (numérico), ocurrencia de llovizna
- `granizo` (numérico), ocurrencia de granizo
- `nieve` (numérico), ocurrencia de nieve
- `t_min_5cm` (numérico), temperatura mínima a intemperie a 5cm en grados centígrados
- `t_min_50cm` (numérico), temperatura mínima a intemperie a 50cm en grados centígrados
- `t_suelo_5cm` (numérico), temperatura media del suelo a 5cm en grados centígrados
- `t_suelo_10cm` (numérico), temperatura media del suelo a 10cm en grados centígrados
- `heliofania_efec` (numérico), heliofanía efectiva en horas
- `heliofania_rel` (numérico), heliofanía relativa en porcentaje
- `p_vapor` (numérico), tensión de vapor en hPa
- `hr` (numérico), humedad relativa en porcentaje
- `td` (numérico), temperatura de rocío en grados centígrados
- `rocio` (numérico), ocurrencia de rocío
- `viento_10m` (numérico), viento a 10 metros en km/h
- `viento_2m` (numérico), viento a 2 metros en km/h
- `rad` (numérico), radiación en MJ/m2
- `etp` (numérico), evapotranspiración en milímetros

Value

Devuelve un data.frame con tantas filas como líneas en el o los archivos leidos y las 25 variables presentes.

See Also

`metadatos_nh()` devuelve los metadatos de las estaciones meteorológicas.

Examples

```
archivo <- system.file("extdata", "NH0358.DAT", package = "agroclimatico")
datos <- leer_nh(archivo)
```

---

leer_surfer *Lee escalas de Surfer*

Description

Lee archivos en el formato "Level File Format" de Surfer (ver http://surferhelp.goldensoftware.com/topics/level_file_format.h

Usage

```
leer_surfer(archivo, color = c("primario", "secundario"))
```

Arguments

`archivo`	ruta al archivo a laeer.
`color`	caracter que indica qué color leer. Puede ser "primario" (que corresponde a "FFGColor") o "secundario" (que corresponde a "FBGColor"). No tiene efecto si el archivo es formato LVL1

Value

Si el archivo es LVL1, un vector con los niveles. Si el archivo es LVL2 o LVL3, una lista con elementos:

- `niveles` (numérico), el nivel a que corresopnde cada color.
- `colores` (caracter), la representación hexadecimal del color de cada break.
- `paleta` (función), una función que toma un entero `n` y devuelve un vector de caracter con `n` colores interpolados a partir de los colores de la escala.

Examples

```
escala <- system.file("extdata", "escala_pp_mensual.lvl", package = "agroclimatico")

escala_pp_mensual <- leer_surfer(escala)

# Valores a los que corresponde cada color
escala_pp_mensual$niveles

# Ver los colores
scales::show_col(escala_pp_mensual$colores)

# Obtener más colores usando la misma paleta
muchos_colores <- escala_pp_mensual$paleta(25)
scales::show_col(muchos_colores)
```

`mapa_argentina` *Mapas*

Description

Mapas de Argentina, sus provincias, departamentos y paises limítrofes. Los mapas de departamentos surgen del repositorio público del Instituto Geográfico Nacional, mientas que el mapa de países limítrofes, Argentina y sus provincias son parte del repositorio Natural Earth.

Usage

```
mapa_argentina()

mapa_provincias(provincias = NULL, departamentos = FALSE)

mapa_argentina_limitrofes()

mapa_departamentos(provincias = NULL)
```

Arguments

`provincias`	vector de caracteres con los nombres de provincias a filtrar. Si es `NULL`, devuelve todas las provincias de Argentina.
`departamentos`	lógico. Si es `TRUE` grafica los departamentos.

Details

Los nombres de las provincias e `mapa_provincias()` son: Buenos Aires, Catamarca, Chaco, Chubut, Ciudad de Buenos Aires, Corrientes, Córdoba, Entre Ríos, Formosa, Islas Malvinas (geometria separada de Tierra del fuego), Jujuy, La Pampa, La Rioja, Mendoza, Misiones, Neuquén, Río Negro, Salta, San Juan, San Luis, Santa Cruz, Santa Fe, Santiago del Estero, Tierra del Fuego y Tucumán y se pueden utilizar para graficarlas individualmente.

Value

Devuelve una tibble con las variables necesarias para generar un mapa utilizando ggplot2 y sf.

Examples

```
library(ggplot2)

# Solo Argentina
ggplot() +
  geom_sf(data = mapa_argentina())

# Argentina y sus provincias
ggplot() +
  geom_sf(data = mapa_provincias())

# Algunas provincias
ggplot() +
  geom_sf(data = mapa_provincias(provincias = c("La Pampa", "Córdoba")))

# Algunas provincias y sus departamentos
ggplot() +
  geom_sf(data = mapa_provincias(provincias = c("La Pampa", "Córdoba"),
                                 departamentos = TRUE))
```

`mapear` *Grafica variables en Argentina*

Description

Dadas mediciones de una variable en puntos ubicados en Argentina, interpola al resto del territorio usando kriging y grafica con contornos. Las funciones secundarias `coord_argentina()` y `theme_inta_mapa()` permiten generar un mapa en la región de Argentina (definida por `xlim` y `ylim`) con el estilo específico usando por INTA.

Usage

```
mapear(
  data,
  valor,
  lon,
  lat,
  breaks = waiver(),
  escala = scales::viridis_pal(),
  cordillera = FALSE,
  variable = NULL,
  titulo = NULL,
  subtitulo = NULL,
```

```
  fuente = NULL
)

coord_argentina(xlim = c(-77, -50), ylim = c(-57, -20), ...)

theme_inta_mapa(...)
```

Arguments

data	data.frame o similar con las variables a utilizar.
valor	vector con los valores medidos.
lon, lat	vectores de ubicación en longitud y latitud.
breaks	vector numérico que define para que valores se graficará los contornos. Si esNULL hace 10 contornos calculados a partir el rango de los datos.
escala	paleta de colores a usar. Tiene que ser una función que reciba un número y devuelva esa cantidad de colores. Por ejemplo escala_temp_min.
cordillera	valor lógico indicando si hay que tapar los datos donde está la cordillera (donde el kriging es particularmente problemático). Si es TRUE pinta con gris donde la altura de la topografía es mayor a 1500 m. También puede ser un número, indicando el valor mínimo desde donde empezar a graficar la cordillera.
titulo, subtitulo, fuente, variable	texto para usar como título, subtítulo, epígrafe y nombre de la leyenda.
xlim, ylim	límites en longitud y latitud.
...	otros argumentos que se pasan a ggplot2::coord_sf() o ggplot2::theme_linedraw().

Value

Un objeto ggplot2.

Examples

```
## Not run:

library(dplyr)

data(datos_nh_mensual)

abril <- datos_nh_mensual %>%
  filter(mes == unique(mes)[4]) #datos del cuarto mes en la base, abril.

abril %>%
  mapear(precipitacion_mensual, lon, lat, cordillera = TRUE,
              escala = escala_pp_mensual,
              titulo = "Precipitación en abril de 2019",
              fuente = "Fuente: INTA",
              variable = "pp")

## End(Not run)
```

`metadatos_nh` *Tabla de Metadatos de Estaciones NH*

Description

Devuelve los metadatos de estaciones incluyendo el código único, ubicación (latitud y longitud) y el nombre.

Usage

```
metadatos_nh(
  codigo = NULL,
  provincia = NULL,
  organismo = NULL,
  lat = NULL,
  lon = NULL
)
```

Arguments

`codigo`, `provincia`, `organismo`	carácter o vector de caracteres para filtrar según código de estación, provincia u organismo.
`lat`	vector numérico con las latitudes límite de la región de interés.
`lon`	vector numérico con las longitudes límite de la región de interés (entre -180 y 180).

Details

Esta función por defecto devuelve la lista completa de estaciones pero alternativamente se puede devolver estaciones específicas a partir de sus códigos o todas las estaciones incluidas en una región. Además incluye un método plot para visualizar rápidamente la ubicación de las estaciones.

Value

data.frame con los metadatos de las estaciones cuyas columnas incluye:

- `codigo_nh` (caracter), variable llave para acceder a los metadatos de estaciones
- `estacion` (caracter), nombre de la estación
- `provincia` (caracter), provincia donde se encuentra la estación
- `organismo` (caracter), organismo a cargo de la estación
- `lat` (numérico), longitud
- `lon` (numérico), latitud
- `altura` (numérico), altura sobre el nivel del mar de la estación

Examples

```
# listado completo de estaciones
head(metadatos_nh(), n = 10)

# listado de estaciones específicas
metadatos_nh(codigo = c("0001", "0011"))

# Filtrar por provincias
metadatos_nh(provincia = c("La Pampa", "Catamarca"))

# Filtrar por organismo
metadatos_nh(organismo = "INTA")

# listados de estaciones en una región
metadatos_nh(lat = c(-30, -20), lon = c(-65, -55))

# gráfico
plot(metadatos_nh())
```

---

olas *Olas*

Description

Identifica periodos de persistencia de un evento definido a partir de alguna condición lógica, por ejemplo días consecutivos donde la temperatura mínima fue igual o menor a 0°C para calcular días acumulados de heladas.

Usage

```
olas(fecha, ..., remplaza.na = FALSE)
```

Arguments

`fecha`	vector de fechas, la serie temporal debe estar completa, sin datos faltantes implicitos.
`...`	umbral o umbrales a calcular utilizando operadores lógicos.
`remplaza.na`	lógico. Por defecto es FALSE, es decír que si la función encuentra un dato faltante "corta" la ola o periodo de persitencia. Si es TRUE, la función reemplaza cada NA por el valor previo en la serie, por lo tanto la ola no se interrumpe si hay NAs.

Details

La función Puede utilizarse en el contexto de `dplyr::summarise()` y `dplyr::group_by()` para hacer este cálculo por grupos.

Value

Devuelve un data.frame con 3 variables fijas y las posibles variables asociadas al agrupamiento:

- `ola` (caracter) nombre de la ola definido por el usuario (si los argumentos de `...` no tienen nombre, se usa `V1`, `V2`, etc...)
- `inicio` (fecha) fecha de inicio de la ola o periodo de persistencia
- `fin` (fecha) fecha de finalización de la ola o periodo de persistencia
- `duracion` (diferencia de fechas, tipo drtn) duración de la ola

Si una ola todavía no terminó, fin y longitud son NA.

Examples

```
data(NH0358)

library(dplyr)
NH0358 %>%
  reframe(olas(fecha, calor = t_max > 20)) %>%
  slice_head(n = 10)

NH0358 %>%
  reframe(olas(fecha, frio = t_min <= 0)) %>%
  slice_head(n = 10)
```

---

pdsi *Índice de Severidad de Sequía de Palmer*

Description

Calcula el Indice de Severidad de Sequía de Palmer. `psdi_ac()` calcula la versión autocalibrada.

Usage

```
pdsi(precipitacion, etp, cc = 100, coeficientes = pdsi_coeficientes())

pdsi_ac(precipitacion, etp, cc = 100, coeficientes = pdsi_coeficientes())
```

Arguments

`precipitacion`	serie de precipitación sin datos faltantes (en mm). Ver sección Details.
`etp`	serie de evapotranspiración potencial sin datos faltantes. Ver sección Details.
`cc`	capacidad de campo (en mm).
`coeficientes`	lista de coeficientes que devuelve `pdsi_coeficientes()`

Details

El Índice de Severidad de Sequía de Palmer, propuesto por Palmer (1965) es usado como indicador para cuantificar las condiciones de sequía a largo plazo. El cálculo usa constantes definidas empíricamente, originalmente utilizando datos meteorológicos de Kansas y de Iowa en Estados Unidos. Estas constantes no representan necesariamente cualquier región del planeta por lo que puede ser redefinidas para el cálculo del índice usando la función del índice usando la función `pdsi_coeficientes()`.

Alternativamente, Wells et al. (2004) propuso el Indice de Severidad de Sequía de Palmer Autocalibrado que tiene la capacidad de ajustar las constantes empíricas durante el cálculo del indice.

Si bien el cálculo de este indice funcionará para series de datos cortas (datos diarios para un mes o datos mensuales para un año), es necesario contar con una climatología, es decir 30 años para que el resultado sea confiable.

Value

Un vector de la misma longitud que `precipitacion` con el PDSI correspondiente a cada caso.

References

Palmer (1965), Meteorological Drought. U.S Weather Bureau, Washington, D.C. (book).

Wells et. al. (2004), A Self-Calibrating Palmer Drought Severity Index. Journal of Climate doi:10.1175/15200442(2004)017<2335:ASPDSI>2.0.CO;2

Examples

```
library(dplyr)

# datos aleatorios
set.seed(42)

datos <- data.frame(fecha = seq(as.Date("1985-01-01"), as.Date("2015-12-01"), by = "1 month"))
datos |>
  mutate(pp = rgamma(nrow(datos), shape = 2, scale = 10),
         etp = rgamma(nrow(datos), shape = 1, scale = 3),
         pdsi_ac = pdsi(pp, etp)) |>
  slice_head(n = 10)
```

---

pdsi_coeficientes *Coeficientes de características climáticas*

Description

Funcion que devuelve los coeficientes de características climáticas necesarios para calcular el Índice de Severidad de Sequía de Palmer con `pdsi()` o su versión autocalibrada `pdsi_ac()`.

Usage

```
pdsi_coeficientes(
  p = 0.897,
  q = 1/3,
  K1.1 = 1.5,
  K1.2 = 2.8,
  K1.3 = 0.5,
  K2 = 17.67
)
```

Arguments

p, q	factores de duración
K1.1, K1.2, K1.3, K2	coeficientes de características climáticas

Details

El cálculo usa constantes definidas empíricamente, originalmente utilizando datos meteorológicos de Kansas y de Iowa en Estados Unidos. Estas constantes no representan necesariamente cualquier región del planeta por lo que puede ser redefinidas para el cálculo del índice usando la función del índice usando la función `pdsi_coeficientes()`.

Value

Una lista con los coeficientes climáticos.

References

Palmer (1965), Meteorological Drought. U.S Weather Bureau, Washington, D.C. (book).

Wells et. al. (2004), A Self-Calibrating Palmer Drought Severity Index. Journal of Climate doi:10.1175/15200442(2004)017<2335:ASPDSI>2.0.CO;2

Examples

```
library(dplyr)

# datos aleatorios
set.seed(42)

datos <- data.frame(fecha = seq(as.Date("1985-01-01"), as.Date("2015-12-01"), by = "1 month"))
datos |>
  mutate(pp = rgamma(nrow(datos), shape = 2, scale = 10),
         etp = rgamma(nrow(datos), shape = 1, scale = 3),
         pdsi_ac = pdsi(pp, etp, coeficientes = pdsi_coeficientes(p = 0.9, q = 1/3))) |>
  slice_head(n = 10)
```

scale_fill_inta *Escalas de colores para precipitación y temperatura*

Description

Escalas para `color` y `fill` para variables discretas.

Usage

```
scale_fill_inta(
  escala,
  name = waiver(),
  breaks = waiver(),
  drop = waiver(),
  ...
)

scale_color_inta(
  escala,
  name = waiver(),
  breaks = waiver(),
  drop = waiver(),
  ...
)
```

Arguments

`escala`	escala de colores. Puede ser • una lista con elementos `niveles` y `paleta` (ver `[leer_surfer()]` y escala_temp_min). • una función que toma un entero `n` y devuelve un vector de caracter con `n` colores interpolados a partir de los colores de la escala.)
`name`	nombre de la escala.
`breaks`	niveles de la escala. Si no es `waiver()`, tiene prioridad por sobre los niveles definidos en `escala`.
`drop`	lógico que indica si se muestran todos los valores o sólo los presentes en los datos. Por defecto, es `FALSE` si la escala define los niveles usando `breaks` o `escala`.
`...`	otros argumentos que se pasan a `ggplot2::scale_fill_manual()` o `ggplot2::discrete_scale()`.

Value

objeto ggproto compatible con ggplot2.

Examples

```
library(ggplot2)
library(dplyr)

pp_enero <- datos_nh_mensual |>
  filter(mes == unique(mes)[1])

# Los contornos llenos requieren que los datos estén en una grilla
# regular, necesitamos hacer una interpolación con kriging.
with(pp_enero, agroclimatico:::kringe(precipitacion_mensual, lon, lat)) |>
ggplot(aes(lon, lat)) +
 geom_contour(aes(z = var1.pred)) +
 geom_contour_filled(aes(z = var1.pred)) +
 scale_fill_inta(escala = escala_pp_mensual)
```

---

spi_indice *Calcula el SPI*

Description

Calcula el Índice Estandarizado de Precipitación para distintas escalas. Las funciones `spi_indice` y `spei_indice` usan internamente a la función SPEI::spi pero tienen la ventaja de devolver el resultado como un data.frame que se puede usar de manera directa para el análisis de datos con dplyr.

Usage

```
spi_indice(
  fecha,
  precipitacion,
  escalas,
  referencia = rep(TRUE, length(fecha)),
  distribucion = "Gamma",
  ...
)

spei_indice(fecha, balance, escalas, distribucion = "log-Logistic", ...)

spi_referencia(fecha, precipitacion)
```

Arguments

`fecha`	vector de fechas.
`precipitacion`	vector de precipitacion.
`escalas`	vector numérico con las escalas requeridas. La unidad de la escala está dada por el vector de fechas. Si `escalas = 6` y los datos son mensuales entonces el cálculo del indice se hará en escalas de 6 meses.

`referencia` serie de precipitación para usar de referencia en el ajuste a la distribución teórica. Puede ser:

- vector lógico o numérico que se usará para filtrar los datos de entrada.
- un data frame con columna `fecha` y `precipitacion`. La función `spi_referencia()` es un simple wrapper a `data.frame()` que le pone el nombre correcto a las variables.

`distribucion` distribución usada para ajustar los datos.

`...` argumentos pasados a SPEI::spi

`balance` balance entre precipitación y evapotranspiración potencial.

Details

La función `spi_indice` toma valores de precipitación mientras que `spei_indice` toma valores del balance entre precipitación y evapotranspiración potencial. Internamente hacen lo mismo; la única diferencia es la distribución teórica usada por defecto para ajustar los datos.

Value

Un data.frame con columnas

- `fecha` (fecha)
- `escala` (numérico) definidas en el argumento de entrada
- `spi` o `spei` (numérico)

References

Vicente-Serrano, S. M., Beguería, S. and López-Moreno, J. I.: A multiscalar drought index sensitive to global warming: The standardized precipitation evapotranspiration index, J. Clim., 23(7), doi:10.1175/2009JCLI2909.1, 2010.

R Package SPEI: Calculation of the Standardized Precipitation-Evapotranspiration Index

Examples

```
library(dplyr)
data(NH0358)

datos_mensuales <- NH0358 %>%
  group_by(fecha = lubridate::round_date(fecha, "month")) %>%
  reframe(precip = mean(precip, na.rm = TRUE),
          etp = mean(etp, na.rm = TRUE))

# Para escalas de 1 a 12 meses
datos_mensuales %>%
  reframe(spi_indice(fecha, precip, escalas = 1:12)) %>%
  slice_head(n = 10)

# Si tenemos nuevos datos y hay que calcular el spi nuevamente pero sin que
# cambien los valores previos, hay que usar `referencia`, por ejemplo usando
# los datos desde el comienzo de la seria hasta 2016
```

```
# Usando un vector lógico
datos_mensuales %>%
  reframe(spi_indice(fecha, precip, escalas = 1:12,
                     referencia = data.table::year(fecha) < 2016)) %>%
  slice_head(n = 10)

# O un data.frame
datos_2016 <- datos_mensuales %>%
 filter(data.table::year(fecha) < 2016)

datos_mensuales %>%
reframe(spi_indice(fecha, precip, escalas = 1:12,
                   referencia = spi_referencia(datos_2016$fecha, datos_2016$precip))) %>%
  slice_head(n = 10)
```

`umbrales` *Calcula la Ocurrencia de eventos a Partir de Umbrales*

Description

La función `umbrales()` permite contar la ocurrencia de eventos definidos a partir de uno o más umbrales.

Usage

```
umbrales(...)
```

Arguments

`...`	umbral o umbrales a calcular utilizando operadores lógicos.

Details

Debe utilizarse en el contexto de `dplyr::summarise()` y opcionalmente `dplyr::group_by()`. Esto permite calcular distintos umbrales y obtener resultados para distintos grupos.

Value

La función devuelve un data.frame con 4 variables fijas junto a posibles variables asociadas a los agrupamientos.

Variables fijas

- `extremo` (caracter) nombre del extremo definido por el usuario (si los argumentos de `...` no tienen nombre, se usa `V1`, `V2`, etc...)
- `N` (numérico) ocurrencia del evento
- `prop` (numérico) proporción de eventos respecto del total de observaciones
- `na` (numérico) proporción de datos faltantes respecto del total de observaciones

Examples

```
data(NH0358)
library(dplyr)

# Sin agrupar devuelve un único valor
NH0358 %>%
 summarise(umbrales(t_30 = t_max >= 30))

# Si se agrupan los datos devuelve un valor por cada grupo
NH0358 %>%
  group_by(fecha = lubridate::floor_date(fecha, "1 month")) %>%
  summarise(umbrales(t_30 = t_max >= 30))

# Se pueden calcular varios umbrales al mismo tiempo
NH0358 %>%
 reframe(umbrales(t_30 = t_max >= 30,
                  t_0  = t_min <= 0))
```

Index