Package 're'

September 6, 2023

Type Package

Title calculating climate scaling factors for the ICBM SOC modell

Version 0.1.0

Author Lorenzo Menichetti

Maintainer Lorenzo Menichetti <ilmenichetti@gmail.com> <lorenzo.menichetti@slu.se>

Description Calculates the climatic scale factors for the ICBM model. The package is a collection of several functions wrapped together.

License , http://www.wtfpl.net

Encoding UTF-8

LazyData true

Depends zoo, lubridate, imputeTS, RColorBrewer, anytime

RoxygenNote 7.2.3

R topics documented:

`add.years` *Extrapolates future climate*

Description

This function extrapolate future climate based on the average of the past 10 years (or less if less are presente) averaging day by day

Usage

```
add.years(dataframe, new.years)
```

Arguments

`dataframe`	the weather data frame
`new.years`	the number of years you want to extend the weather data frame

Value

a table, same structure of template

Author(s)

Lorenzo Menichetti <ilmenichetti@gmail.com>

Examples

```
add.years(dataframe, 5)
```

`FC` *internal function for determining the soil field capacity*

Description

internal function for determining the soil field capacity

Usage

```
FC(sand, SOC)
```

Arguments

`sand`	sand %
`SOC`	SOC %

Value

a single numerical value with the soil field capacity

Author(s)

Lorenzo Menichetti <ilmenichetti@gmail.com>

References

Tóth, B., M. Weynants, A. Nemes, A. Makó, G. Bilas, and G. Tóth. 2015. "New Generation of Hydraulic Pedotransfer Functions for Europe: New Hydraulic Pedotransfer Functions for Europe." European Journal of Soil Science 66 (1): 226–38. https://doi.org/10.1111/ejss.12192.

`fill.na` *Fill NAs in the data with linear interpolation*

Description

This function fills NAs based on the average of the past 10 years (or less if less are presente) averaging day by day

Usage

```
fill.na(dataframe)
```

Arguments

`dataframe`	the weather data frame

Value

a table, same structure of dataframe

Author(s)

Lorenzo Menichetti <ilmenichetti@gmail.com>

Examples

```
add.years(dataframe, 5)
```

GAI *Internal function to calculate the GAI (Green Area Index) based on yields and agronomical data.*

Description

The function distributes the biomass growth (considering its proxy GAI and LAI) according to a gaussian function with parameters defined relatively to the crop. It is used for determining the effect of increased ET due to the presence of plant and the effect of shading on soil temperature.

Usage

```
GAI(
  yield,
  crop,
  year,
  variance,
  seeding,
  harvest,
  tillage,
  minimum_cover,
  yield2 = NULL,
  harvest2 = NULL
)
```

Arguments

`yield`	annual yields (kg/ha)
`crop`	crop id, either "spring_small_grains", "spring_oil_seeds","winter_small_grains", "winter_oil_seeds","root_crop", "fodder", "fodder maize" or "ley"
`year`	sequence of the years to run the simulation for
`variance`	the variance of the gaussian used to simulate the GAI, in days
`seeding`	day of seeding (day number of the year)
`harvest`	day of harvest (day number of the year)
`tillage`	day of tillage (day number of the year)
`minimum_cover,`	forcing a minimum GAI cover, useful for leys (dimensionless)
`yield2`	OPTIONAL, in case there is more than one harvest per year (kg/ha)
`harvest2`	OPTIONAL, in case there is more than one harvest per year (days)

Details

The function relies on an input matrix which must follow a precise format, please refer to the attached template.

The function is used to simulate the development of crops and their green area index (G.A.I.). The function uses among the inputs a vector of different crops, which will be simulated with different parameters. These are `spring_small_grains`, `spring_oil_seeds`, `winter_small_grains`, `winter_oil_seeds`,`root_crop`, `fodder`, `fodder maize` and `ley`. The function loops in annual steps through the years of the simulation and then runs a nested loop to simulate the crop growth in daily steps. The function initially calls the is.leapyear function to decide if to use 365 or 366 days in the simulation. Different crops are simulated in different ways. The functions returns also the LAI, calculated as

$$0.8 \cdot GAI$$

until maximum GAI. After maximum GAI, before harvest the LAI is set to never fall below

$$0.7 \cdot max(LAI)$$

, and between harvest and tillage never below

$$0.2 \cdot max(LAI)$$

according to Kätterer & Andrén 2009.

First of all the function checks if the crop of that year is not "fodder", "fodder maize" or "ley". if not, then sets the maximum GAI (j is the index used in the main function loop, looping through the simulation years):

$$GAI_{max} = 0.0129 \cdot (\frac{yield_j}{1000})^2 + 0.1964 \cdot (\frac{yield_j}{1000})$$

For root crops the maximum GAI is set differently (see below the specific section) The function then proceeds to simulate the growth according to a gausssian function subsequently modified. The gaussian function is controlled by the parameters defining where it is centered and its variance. Its center is calculated according to seeding and harvest dates, which are in the input data. Then the GAI outside the area covered by such function is either set to zero or to the minimum coverage specified in the input data. The main function used to simulate the crop growth, after first having calculated the center of the gausian with

$$middle = seeding_j + \frac{harvest_j - seeding_j}{2}$$

is the following:

$$GAI = GAI_{max} \cdot exp(-\frac{(day - middle)^2)}{(2 \cdot variance_j)})$$

Most crops are considered covering the soil even after being fully mature, except root crops fodder (including silage maize). Please not that this does not imply that such crops are returned as C inputs to the soil in the ICBM model, this concerns just the calculation of the climatic reduction coefficients.

Exceptions:

`root_crops`*:*
Root crops have a specific function, which is based on the average yields (yield_vec) and maximum LAI (LAI_max_vec) obtained in the Ultuna experiment during the three years when root crops were planted. The maximum GAI is also calculated with a different function:

$$GAI_{max} = min(5.6, \frac{1}{0.8} \cdot mean(\frac{LAI_max_vec}{yield_vec}) \cdot \frac{1}{0.75} \cdot yield_j)$$

`fodder`*:*
The maximum GAI is calculated according to data for fodder rape (https://www.agronomysociety.org.nz/files/2010_1

$$GAI_{max} = min(10, 0.0004615385 \cdot yield_j)$$

`fodder_maize`*:*
The maximum GAI is calculated according to data from the Ultuna experiment, where silage maize has been planted since 2000

$$GAI_{max} = min(10, \frac{1}{0.8} \cdot 0.000533 \cdot yield_j)$$

`ley`*:*
Leys are complicated by the fact that there might be two subsequent cuts, so two harvests. The command considers this possibility with 2 optional parameters, harvest2 and yields2, which are otherwise set to NULL. If these two parameters are present another if condition takes care of them when they are not set to zero.

Value

A data frame of 5 variables: `date`, `GAI`, `crop`, `yields_at_harvest` and `LAI`. The other dimension of the data frame is as long as the combination of the treatments and the days of the simulation.

Author(s)

Lorenzo Menichetti <ilmenichetti@gmail.com>

References

W. Mazurczyk, Anna Teresa Wierzbicka, Anna Teresa Wierzbicka, C. Trawczyński,2009,. HARVEST INDEX OF POTATO CROP GROWN UNDER DIFFERENT NITROGEN AND WATER SUPPLY.

T. Kätterer and O. Andrén, "Predicting daily soil temperature profiles in arable soils in cold temperate regions from air temperature and leaf area index," Acta Agric. Scand. Sect. B - Plant Soil Sci., vol. 59, no. 1, pp. 77–86, 2009, doi: 10.1080/09064710801920321.

Examples

```
data(aboveground_testdata) #load the example dataset

selected_aboveground<-aboveground_testdata[aboveground_testdata$treat=="CONVENTIONAL",]

GAI_test<-GAI(yield=selected_aboveground$total_dm_kg_ha, crop=selected_aboveground$crop_id,
       year=selected_aboveground$year, variance=selected_aboveground$variance,
       seeding=selected_aboveground$seeding, harvest=selected_aboveground$harvest,
     tillage=selected_aboveground$tillage, minimum_cover=selected_aboveground$minimum_cover)

plot(GAI_test$date, GAI_test$GAI, type="l") # plotting the results to test
```

`is.leapyear` *Identifies if a year is leap or not*

Description

Well... pretty simple one, main heading is self-explanatory

Usage

```
is.leapyear(year)
```

Arguments

`year`	a year

Author(s)

Lorenzo Menichetti <ilmenichetti@gmail.com>

Examples

```
is.leapyear(1996)
```

`poros` *Internal function for determining the soil porosity from texture.*

Description

If clay is present the function uses Toth et al., 2015, otherwise Kätterer et al., 2006

Usage

```
poros(sand, clay, SOC)
```

Arguments

`sand`	sand content%
`clay`	clay content % (optional)
`SOC`	SOC content %

Value

a single numerical value with the soil porosity

Author(s)

Lorenzo Menichetti <ilmenichetti@gmail.com>

References

Tóth, B., M. Weynants, A. Nemes, A. Makó, G. Bilas, and G. Tóth. 2015. "New Generation of Hydraulic Pedotransfer Functions for Europe: New Hydraulic Pedotransfer Functions for Europe." European Journal of Soil Science 66 (1): 226–38. https://doi.org/10.1111/ejss.12192. Kätterer, T., O. Andrén, and P-E. Jansson. 2006. "Pedotransfer Functions for Estimating Plant Available Water and Bulk Density in Swedish Agricultural Soils." Acta Agriculturae Scandinavica, Section B - Plant Soil Science 56 (4): 263–76. https://doi.org/10.1080/09064710500310170.

reclim *Wrapper for the functions calculating the ICBM climate scaling factors*

Description

This functions runs the re_clim calculation on a dataset (composed by two different tables, one daily for weather and one annual for aboveground biomass, please refer to the template for the data structure) The function is a wrapper, performing a few data checks and running functions to calculate several parameters and hopefully runs without the user having to bother too much with intermediate steps.

Usage

```
reclim(
  weather,
  aboveground,
  sun.mode,
  latitude,
  altitude,
  depth,
  sand = NULL,
  clay = NULL,
  ave_SOC = NULL,
  porosity = NULL,
  wilting_point = NULL,
  field_capacity = NULL
)
```

Arguments

`weather`	data matrix of weather data, must be exactly in the format of the template attached as example and contain the following headers: ("date", "year", "month", "day", "air_temp_deg_C", "precipitation_mm", "windspeed_kmh", "humidity_percent", "Rsolar_lang")
`aboveground`	data matrix of weather data, must be exactly in the format of the template attached as example and contain the following headers: ("year", "crop_description", "crop_id", "treat", "variance", "seeding", "harvest", "harvest2", "tillage", "minimum_cover", "total_dm_kg_ha", "total_dm_kg_ha2") "harvest2" and "total_dm_kg_ha2" are optional and used in case of a double cut for leys
`sun.mode`	mode of sun data, can be either "Rsolar" (expressed in Langleys) or "cloudiness" (expressed in percent of sunny time per day)
`latitude`	the latitude, in degrees
`altitude`	altitude in meters
`depth`	depth considered in centimeters
`sand`	sand, in %. This is needed if porosity, wilting point and field capacity are not specified
`clay`	clay, in %. This is needed if porosity, wilting point and field capacity are not specified

ave_SOC	average SOC over the whole period, in %. This is needed if porosity, wilting point and field capacity are not specified
porosity	soil porosity, as 0 to 1. If speciefied with wilting point and field capacity there's no need for other soil edaphic properties.
wilting_point	wilting point, as mm over the total. If speciefied with porosity and field capacity there's no need for other soil edaphic properties.
field_capacity	field capacity , as mm over the total. If speciefied with wilting point and porosity there's no need for other soil edaphic properties.

Author(s)

Lorenzo Menichetti <ilmenichetti@gmail.com>, Martin Bolinder, Olaf Andrén, Thomas Kätterer

Examples

```
reclim_out<-reclim(weather=weather_testdata,
                   aboveground=aboveground_testdata,
                   latitude=44,
                   altitude=20,
                   sand=22,
                   clay=36,
                   ave_SOC=1.2,
                   depth=20,
                   sun.mode="Rsolar")
```

---

re_temperature — *internal function for determining the dependence of decomposition over soil temperature*

Description

internal function for determining the dependence of decomposition over soil temperature

Usage

```
re_temperature(soilT)
```

Arguments

soilT	soil temperature daily values

Details

in case of NAs in the inputs, the function attempts to fill them with a Stineman interpolation

Value

a vector with the daily water reduction values

Author(s)

Lorenzo Menichetti <ilmenichetti@gmail.com>

re_water	*internal function for determining the dependence of decompositono over soil moisture*

Description

internal function for determining the dependence of decompositono over soil moisture

Usage

```
re_water(twilt, tfield, water, porosity, L)
```

Arguments

`twilt`	wilting point (0 to 1)
`tfield`	field capacity (0 to 1)
`water`	water balance (in mm)
`porosity`	soil porosity /0 to 1)
`L`	soil depth (in mm)

Value

a vector with the daily water reduction values

Author(s)

Lorenzo Menichetti <`ilmenichetti@gmail.com`>

References

Moyano FE, Manzoni S, Chenu C. Responses of soil heterotrophic respiration to moisture availability: An exploration of processes and models. Soil Biol Biochem. Elsevier Ltd; 2013;59: 72–85. doi:10.1016/j.soilbio.2013.01.002

soiltemp	*internal function for determining the soil temperature from the air temperature*

Description

internal function for determining the soil temperature from the air temperature

Usage

```
soiltemp(L, GAI, date, temperature, LAI = NULL)
```

Arguments

`L`	soil depth (mm)
`GAI`	green area index daily values
`date`	date vector (daily steps)
`temperature`	air temperature (°C)
`LAI`	(optional) LAI. If not present LAI is calculated only according to LAI=0.8*GAI, otherwise LAI is used directly

Details

The function calculates first the surface temperature. If the temperature is below zero:

$$T_{surface_i} = 0.20 \cdot T$$

And if the temperature is above zero

$$T_{surface_i} = T_i \cdot (0.95 + 0.05 \cdot exp(-0.4 \cdot (LAI_i - 3))$$

And then calculates the soil temperature according to:

$$T_{soil_{i+1}} = T_{soil_i} + (T_{surface_i} - T_{soil_i}) \cdot 0.24 \cdot e^{(-Z_{depth} \cdot 0.017)} \cdot exp(-0.15 \cdot GAI_i)$$

And where

$$Z_{depth} = \frac{L}{20}$$

The LAI is calculated as \dfuncLAI= 0.8 \cdot GAI

Value

a vector with the daily soil temperature values

Author(s)

Lorenzo Menichetti <`ilmenichetti@gmail.com`>

References

Kätterer, T., and O. Andrén. 2009. "Predicting Daily Soil Temperature Profiles in Arable Soils in Cold Temperate Regions from Air Temperature and Leaf Area Index." Acta Agriculturae Scandinavica, Section B - Plant Soil Science 59 (1): 77–86. https://doi.org/10.1080/09064710801920321.

`templates` *template data used to test the reclim package*

Description

template data used to test the reclim package

Format

two data frames. Data are fictional, generated from real measured data but with a great deal of noise added. First dataframe (daily climatic values), with 5114 rows and 9 variables:

date date vector, YYYY-mm-dd

year year, integer

month month, integer

day day, integer

air_temp_deg_C mean daily air temperature (°C)

precipitation_mm cumulated daily precipitation, in mm

windspeed_kmh mean daily wind speed, in km/h

humidity_percent mean daily air humidity, in %

Rsolar solar radiation, in this case in Langleys. Can be also cloudiness, in hours of sun per day ...

Second dataframe (annual crop data), with 28 rows and 12 variables:

year date vector, YYYY-mm-dd

crop_description year, integer

crop_id month, integer

treat day, integer

variance variance of the biomass distibution function, in days

seeding seeding day, in day of year

harvest first harves day, in day of year

harvest2 second harvest day (if present, for lay), in day of year

tillage tillage day, in day of year

minimum_cover minimum biomass on the ground all the time, kg of dry mass per ha

total_dm_kg_ha aboveground biomass of first harvest, kg of dry mass per ha

total_dm_kg_ha2 aboveground biomass of second harvest (if present, for lay), kg of dry mass per ha ...

Author(s)

Lorenzo Menichetti <ilmenichetti@gmail.com>

Examples

```
data(aboveground_testdata)
data(weather_testdata)
```

waterbalance *Internal function for the water balance model*

Description

Internal function for the water balance model

Usage

```
waterbalance(twilt, tfield, precipitation, GAI, date, ET0, L, alpha = 0.7)
```

Arguments

`twilt`	wilting point (0 to 1)
`tfield`	field capacity (0 to 1)
`precipitation`	daily precipitations (mm)
`GAI`	gren area index daily values
`date`	date vector
`ET0`	Evapotranspiration (calculated based on PET and GAI)
`L`	soil depth (mm)

Details

The formulas come mainly from Allen et al., 1998 https://www.fao.org/3/x0490e/x0490e00.htm and it is used to simulate the soil water balance. The calculation is done through multiple steps, iterated for each timestep:

Step 1: Soil water W is initialized assuming saturation, based on the depth L and volumetric capacity

$$W[1] = \Theta_f \cdot L$$

Step 2: The single crop coefficient Kc is calculated based on GAI

$$K_c = 1.3 - 0.5 \cdot exp(-0.17 \cdot GAI)$$

Step 3: calculation of crop evapotranspiration (ETc) under standard condition

$$ET_c = ET_0 \cdot K_c$$

Step 4: the intercepted water It is calculated based on crop ET, GAI and precipitation P

$$It = min(P, ET_c, 0.2 \cdot GAI)$$

Step 5: potential evapotraspiration is calculated

$$E_{pot} = (ET_c - It)$$

Step 6: Calculation of the percolation. Water (W_b, water bypass) is lost when above field capacity, but allowing saturation for one day

$$W_b = max(0, W - (\Theta_f \cdot L))$$

Step 7: Soil evaporation reduction coefficient

$$Kr = (1 - (0.95 \cdot tfield - \Theta)/(0.95 \cdot tfield - \alpha \cdot twilt))^2$$

Subsequent conditions are applied so that Kr cannot be above one, and the values before the minimum Kr are also zero. *Step 8: Actual evapotraspiration is calculated*

$$E_{act} = E_{pot} \cdot Kr$$

Step 9: The water balance is calculated (stepwise)

$$W[i+1] = W[i] + P[i] - E_{act}[i] - It - W_b[i]$$

Value

The function returns a data frame with water balance and date (days)

Author(s)

Lorenzo Menichetti <`ilmenichetti@gmail.com`>

`WP` *Internal function for determining the soil wilting point*

Description

Internal function for determining the soil wilting point

Usage

```
WP(sand, clay, SOC)
```

Arguments

`sand`	sand content %
`clay`	clay content % (optional)
`SOC`	SOC content%

Value

a single numerical value with the soil wilting point

Author(s)

Lorenzo Menichetti <`ilmenichetti@gmail.com`>

References

Tóth, B., M. Weynants, A. Nemes, A. Makó, G. Bilas, and G. Tóth. 2015. "New Generation of Hydraulic Pedotransfer Functions for Europe: New Hydraulic Pedotransfer Functions for Europe." European Journal of Soil Science 66 (1): 226–38. https://doi.org/10.1111/ejss.12192.

Index