Package: ZeBook (via r-universe)

September 14, 2024

Type Package

Title Working with Dynamic Models for Agriculture and Environment

Version 1.1

Date 2018-11-08

Author Francois Brun (ACTA), David Makowski (INRA), Daniel Wallach (INRA), James W. Jones (U.of Florida).

Maintainer Francois Brun <francois.brun@acta.asso.fr>

Description R package accompanying the book Working with dynamic models for agriculture and environment, by Daniel Wallach (INRA), David Makowski (INRA), James W. Jones (U.of Florida), Francois Brun (ACTA). 3rd edition 2018-09-27.

License LGPL-3

LazyData yes

Depends R(>= 2.10.0)

Imports triangle, deSolve, stats, graphics

LazyDataCompression xz

RoxygenNote 6.1.1

NeedsCompilation no

Date/Publication 2018-11-09 17:40:03 UTC

Repository https://fbrun-acta.r-universe.dev

RemoteUrl https://github.com/cran/ZeBook

RemoteRef HEAD

RemoteSha b892a7e80a233b1c468526307eb5f7b49d95514d

Contents

`ZeBook-package` *Working with Dynamic Models for Agriculture and Environment*

Description

Package:	ZeBook
Type:	Package
Version:	1.1
Date:	2018-11-08

License:	LGPL-3
LazyLoad:	yes
LazyData:	yes
Depends:	R(>= 2.10.0)
Imports:	triangle, deSolve, stats, graphics

ZeBook Working with dynamic models for agriculture and environment (Working with Dynamic Crop Models)

Linked to book **Working with Dynamic Crop Models** (Elsevier), Third edition, 27 septembre 2018 by Wallach, Makowski, Jones and Brun. http://www.modelia.org/moodle/course/view.php?id=61

A full description of the models is in the book in appendix of the book.

Chapter numbers have changed between Second edition and Third Edition. Here the chapter numbers in the demo were changed to fit to Third edition. But all materials available in Second edition are still available in this version.

ACKNOWLEDGMENTS The project "Associate a level of error in predictions of models for agronomy" (CASDAR 2010-2013) and the French network "RMT modeling and agriculture", http://www.modelia.org) have contributed to the development of this R package. This project and network are lead by ACTA (French Technical Institute for Agriculture) and was funded by a grant from the Ministry of Agriculture and Fishing of France.

Other contributions Juliette Adrian, Master2 internship (ACTA, http://www.modelia.org/moodle/mod/resource/view.php?id=1027), january-jully 2013.

Sylvain Toulet, Master2 internship (INRA, http://www.modelia.org/moodle/mod/resource/view.php?id=965), january-jully 2012.

Author(s)

Francois Brun (ACTA) <francois.brun@acta.asso.fr>, David Makowski (INRA), Daniel Wallach (INRA), James W. Jones (U.of Florida),

References

Working with Dynamic Crop Models (Elsevier), Third edition http://www.modelia.org

---

`AICf` *Calculate AIC, Akaike's Information Criterion*

Description

This function calculate AIC criterion given a vector of observation, a vector of prediction and number of parameter. Note that number of parameters should include variance. AICcomplete is the same calculation of the AIC function of R (AICcomplete = n*log(RSS/n)+n+n*log(2*pi)+2*p, with p including variance). AICshort is the calculation described in chapter 6 Working with crop model (AICshort =n*log(RSS/n)+2*p, with p including variance). difference between AICcomplete and

AICshort is AICcomplete-AICshort=n+n*log(2*pi) As you use AIC to compare models (with different number of parameters) on a same data (with same n, number of observation), you can use AICshort or AICcomplete.

Usage

```
AICf(Yobs, Ypred, npar)
```

Arguments

Yobs	: observed values
Ypred	: prediction values from the model
npar	: number of parameters (should include variance that count for one supplementary parameter)

Value

a vector with AICcomplete and AICshort

Examples

```
x=c(1,2,3,4,5)
y=c(1.2,1.8,3.5,4.3,5.5)
fit = lm(y~x)
AIC(fit)
AICf(y,predict(fit),3) # 3 parameters : intercept, slope and variance
```

Bean *Bean gene-based models dataset*

Description

Genetic data for the common bean (Phaseolus vulgaris L) that was based on a population created by C. E. Vallejos (personal communication; also see Bhakta et al., 2015, 2017) by crossing two widely-differing cultivars of bean (Calima, an Andean type, with Jamapa, a Mesoamerican type). Bean$marker : Bean Marker Data Bean$MET : Weather data on 5 Locations Bean$QTL : QTL Data Bean$modelpar : Dynamic Model parameters

Usage

```
Bean
```

Format

a List including 4 data.frame Bean$marker, Bean$MET, Bean$QTL, Bean$modelpar.

Source

C. E. Vallejos (personal communication) and Bhakta et al. (2015, 2017). Bhakta, M. S., V. A. Jones, C. E. Vallejos. 2015. Punctuated distribution of recombination hotspots and demarcation of pericentromeric regions in Phaseolus vulgaris L. PLoS ONE 10(1): https://doi.org/10.1371/journal.pone.0116822 Bhakta, M. S., S. A. Gezan, J. A. Clavijo Michelangeli, M. Carvalho, L. Zhang, J. W. Jones, K. J. Boote, M. J. Correll, J. Beaver, J. M. Osorno, R. Colbert, I. Rao, S. Beebe, A. Gonzalez, J. Ricaurte, and C. E.do Vallejos, 2017. A predictive model for time-to-flower in the common bean based on QTL and environmental variables. G3: Genes, Genomes, Genetics 7(12) 3901-3912. https://doi.org/10.1534/g3.117.300229.

Examples

```
# show the maker of JC1 to JC9 values for both parents (JAM and CAL)
# and 5 cRILS (RIJC001 to RIJC005)
Bean$marker[2:8,1:10]
# show the first value of weather data
head(Bean$MET)
# show the value of QTL
Bean$QTL[4:10,1:10]
# show the value of
Bean$modelpar
```

`carbonsoil.model` *The CarbonSoil model - calculate daily values over designated time period*

Description

Simple dynamic model of soil carbon content, with a time step of one year. The equations that describe the dynamics of this system are adapted from the Henin-Dupuis model described in Jones et al. (2004). The soil carbon content is represented by a single state variable: the mass of carbon per unit land area in the top 20 cm of soil in a given year (Z, kg.ha-1). It is assumed that soil C is known in some year, taken as the initial year. The yearly change in soil C is the difference between input from crop biomass and loss.

Usage

```
carbonsoil.model(R, b, U, Z1, duration)
```

Arguments

`R`	: the fraction of soil carbon content lost per year
`b`	: the fraction of yearly crop biomass production left in the soil
`U`	: the amount of C in crop biomass production (constant or time series)
`Z1`	: initial soil carbon content
`duration`	: duration of simulation (year))

Value

Soil carbon years.

`carbonsoil.update` *The CarbonSoil model - calculate change in soil carbon for one year*

Description

Simple dynamic model of soil carbon content, with a time step of one year. The equations that describe the dynamics of this system are adapted from the Henin-Dupuis model described in Jones et al. (2004). The soil carbon content is represented by a single state variable: the mass of carbon per unit land area in the top 20 cm of soil in a given year (Z, kg.ha-1). It is assumed that soil C is known in some year, taken as the initial year. The yearly change in soil C is the difference between input from crop biomass and loss.

Usage

```
carbonsoil.update(Zy, R, b, Uy)
```

Arguments

`Zy`	: Soil carbon content for year
`R`	: the fraction of soil carbon content lost per year
`b`	: the fraction of yearly crop biomass production left in the soil
`Uy`	: the amount of C in crop biomass production in the given year

Value

Soil carbon content for year+1.

`carcass.define.param` *Define values of the parameters for the Carcass model*

Description

Define parameters values

Usage

```
carcass.define.param(full = FALSE)
```

Arguments

`full`	: if TRUE, return the full description of distribution(default = FALSE)

Value

matrix with parameter values (nominal, binf, bsup). A data.frame if full=TRUE

Examples

```
carcass.define.param(full=TRUE)
```

`carcass.EMI.model` *The Carcass (growth of beef cattle) model with energy as input*

Description

Model description. This model is proposed by Hoch et. al (2004) to represent the growth of cattle and the relative body composition of diferent types of animals depending on nuritionnal conditions. It simulates the dynamics of changes in the composition of the body fat and proteins according to nutrient intake. The system is represented by four state variables: the protein and fat in the carcass (resp. ProtC and LIPC) and other tissues (resp. ProtNC and LipNC) grouped under the name of the fifth district (again, gastrointestinal tract, skin .. .). These variables depend on time, the time step used is dt = 1 day. The model is defined by 20 equations, with a total of 18 parameters for the described process.

Usage

```
carcass.EMI.model(protcmax, protncmax, alphac, alphanc, gammac, gammanc,
  lip0, lipc1, lipnc1, beta, delta, amW, b0c, b1c, b0nc, b1nc, c0, c1,
  energie, PVi, duration)
```

Arguments

`protcmax`	: amounts of protein in the carcass of the adult animal (kg)
`protncmax`	: amounts of protein in the 5th district of the adult animal (kg)
`alphac`	: maximum protein synthesis rate in the frame (excluding basal metabolism) (j-1)
`alphanc`	: maximum rate of protein synthesis in the 5th district (except basal metabolism) (j-1)
`gammac`	: Maximum rate of protein degradation in the frame (excluding basal metabolism) (j-1)
`gammanc`	: maximum rate of protein degradation in the 5th district (except basal metabolism) (j-1)
`lip0`	: maximum lipid concentration to the theoretical physiological age (percent)
`lipc1`	: increase coefficient of the maximum lipid concentration with the physiological age of the carcase (percent)
`lipnc1`	: increase coefficient of the highest lipid concentration with physiological age area in the 5th (percent)

beta	: lipid synthesis rate (j-1)
delta	: lipid degradation rate (d-1)
amW	:
b0c	: coefficient of the allometric equation linking mass and lipid-protein carcass
b1c	: exponent allometric equation linking mass and defatted protein carcass
b0nc	: coefficient of the allometric equation linking mass and lipid-protein 5th district
b1nc	: exponent allometric equation linking mass and lipid-protein 5th district
c0	: coefficient of the allometric equation between live weight and live weight empty
c1	: exponent allometric equation linking body weight and live weight empty
energie	: Metabolizable energy available
PVi	: initial liveweight
duration	: duration of simulation

Value

matrix with ProtC,LipC,ProtNC,LipNC,PV

---

carcass.EMI.model2 *The Carcass model function for use with carcass.EMI.simule*

Description

see carcass.EMI.model for model description.

Usage

```
carcass.EMI.model2(param, energie, PVi, duration)
```

Arguments

param	: a vector of parameters
energie	: Metabolizable energy available
PVi	: initial liveweight
duration	: duration of simulation

Value

data.frame with PV,ProtC,ProtNC,LipC,LipNC

carcass.EMI.multi — *Wrapper function to run Carcass model on several animals with different conditions*

Description

wrapper function for multisimulation with carcass.EMI.model2

Usage

```
carcass.EMI.multi(param, list_individuals, energy, init_condition)
```

Arguments

`param`	: a vector of parameters
`list_individuals`	: list of individuals
`energy`	: Metabolizable energy available for all individuals
`init_condition`	: initial condition for all individuals

Value

data.frame with id, ration ,duration, day, PV,ProtC,ProtNC,LipC,LipNC

carcass.EMI.simule — *Wrapper function to the Carcass model for multiple sets of parameter values*

Description

Wrapper function to the Carcass model for multiple sets of parameter values

Usage

```
carcass.EMI.simule(X, energy, PVi, duration)
```

Arguments

`X`	: parameter matrix
`energy`	: Metabolizable energy available
`PVi`	: initial liveweight
`duration`	: duration of simulation

Value

data.frame with PV,ProtC,ProtNC,LipC,LipNC

carcass.model *The Carcass (growth of beef cattle) model*

Description

Model description.

The model is defined by 20 equations, with a total of 19 parameters for the described process.

Usage

```
carcass.model(protcmax, protncmax, alphac, alphanc, gammac, gammanc, lip0,
  lipc1, lipnc1, beta, delta, k, b0c, b1c, b0nc, b1nc, c0, c1, cem,
  duration)
```

Arguments

protcmax	: amounts of protein in the carcass of the adult animal (kg)
protncmax	: amounts of protein in the 5th district of the adult animal (kg)
alphac	: maximum protein synthesis rate in the frame (excluding basal metabolism) (j-1)
alphanc	: maximum rate of protein synthesis in the 5th district (except basal metabolism) (j-1)
gammac	: Maximum rate of protein degradation in the frame (excluding basal metabolism) (j-1)
gammanc	: maximum rate of protein degradation in the 5th district (except basal metabolism) (j-1)
lip0	: maximum lipid concentration to the theoretical physiological age (percent)
lipc1	: increase coefficient of the maximum lipid concentration with the physiological age of the carcase (percent)
lipnc1	: increase coefficient of the highest lipid concentration with physiological age area in the 5th (percent)
beta	: lipid synthesis rate (j-1)
delta	: lipid degradation rate (d-1)
k	: Parameter coefficient between the half-saturation of the Michaelis-Menten equation of the metabolic weight (MJ.kg^0.75)
b0c	: coefficient of the allometric equation linking mass and lipid-protein carcass
b1c	: exponent allometric equation linking mass and defatted protein carcass
b0nc	: coefficient of the allometric equation linking mass and lipid-protein 5th district
b1nc	: exponent allometric equation linking mass and lipid-protein 5th district
c0	: coefficient of the allometric equation between live weight and live weight empty
c1	: exponent allometric equation linking body weight and live weight empty
cem	:
duration	: duration of simulation

Value

data.frame with ProtC,LipC,ProtNC,LipNC,PV

`carcass_data` *Data of growth of beef cattle for Carcass model*

Description

This dataset is a list of 5 dataframe. list_individuals : identifiant for each indivual energy : Ration (type ration), Individu, time (week), energie (?) init_condition : Individu, Pvi (initial liveweight) observation_dynamic :Ration (3 levels "C","EM","F"), individu, time, PVobs observation_slaughter : Ration, individu, time, PVVobs, PVobs, ProtCobs, ProtNCobs, LipCobs, LipNCobs.

Usage

```
carcass_data
```

Format

a `RangedData` instance, 1 row per plot.

Source

Agabriel, J. (com.pers.)

`carrot.weevil.model` *Carrot weevil development model*

Description

Model description. Simple model of developpement of carrot weevil.

Usage

```
carrot.weevil.model(tbase = 7, tteggs = 130, ttlarvae = 256,
  ttprepupae = 114, ttpupae = 130, ttadultpreovi = 91, weather,
  sdate = 1, ldate = 360)
```

Arguments

tbase	: base temperature
tteggs	: duration of eggs stage in degre.day
ttlarvae	: duration of larvae stage in degre.day
ttprepupae	: duration of prepupae stage in degre.day
ttpupae	: duration of pupae stage in degre.day
ttadultpreovi	: duration of adult stage until egg laying in degre.day
weather	: weather data.frame for one single year
sdate	: date to begin simulation (day of year) (default 1)
ldate	: date to end simulation (day of year) (default 360)

Value

data.frame with daily state variable

---

chicks_data *Data of growth of chicks*

Description

This dataset content dynamic measurements of growth of chicks for different individuals and different strains. The data comes from a selection experiment on chicken initiated by F. Ricard Research Station on Poultry of INRA Nouzilly. The selection focuses on weight at 8 and 36 weeks and allowed to differentiate the following five strains:

strain 1 : X-+ (low at 8, but high at 36 weeks)

strain 2 : X+- (high at 8, but low at 36 weeks)

strain 3 : X++ (high weight at both ages)

strain 4 : X- (low weight for both ages)

strain 5 : X0 (control).

This is a sub-sample of 50 females in the last generation of selection with weight data (in g) at 12 different ages to complete measurement (0, 4, 6, 8, 12, 16, 20, 24; 28, 32, 36 and 40 weeks).

Usage

```
carcass_data
```

Format

a RangedData instance, 1 row : strain ; id_animal ; time (day) ; liveweight (g).

Source

Duval M., Robert-Granie C., Foulley J.-L. (2009) Estimation of heterogeneous variances in non linear mixed models via the SAEM algorithm with applications to growth curves in poultry. Journal de la Societe Francaise de Statistique, 150,65-83

Donnet S., Foulley J.-L., Samson A. (2010) Bayesian analysis of growth curves using mixed models defined by stochastic differential equations. Biometrics 66, 733-741

Jaffrezic F., Meza C., Foulley .J.-L., Lavielle M. (2006) The SAEM algorithm for the analysis of non linear traits in genetic studies. Genetics, Selection, Evolution,38, 583-

This dataset was used in a training session Biobayes (France, 2011) training session.

Albert I., Ancelet S., David O., Denis J.B., Makowski D., Parent E., Soubeyrand S. (2012) Methodes statistiques bayesiennes. Bases theoriques et applications en alimentation, environnement et genetique. FormaScience. Ecole-chercheurs INRA.

---

cotton.model *The Cotton model (dynamic for numbers of Cotton fruiting points).*

Description

Model description. TO COMPLETE

Usage

```
cotton.model(TESQ, PMAX, AFL, AL, AOP, P1, P2, P3, P4, P5, PF, PSF, TSQ, P,
  PR, PT, tend)
```

Arguments

TESQ	: TO COMPLETE
PMAX	: TO COMPLETE
AFL	: TO COMPLETE
AL	: TO COMPLETE
AOP	: TO COMPLETE
P1	: TO COMPLETE
P2	: TO COMPLETE
P3	: TO COMPLETE
P4	: TO COMPLETE
P5	: TO COMPLETE
PF	: TO COMPLETE
PSF	: TO COMPLETE
TSQ	: TO COMPLETE
P	: TO COMPLETE
PR	: TO COMPLETE
PT	: TO COMPLETE
tend	: TO COMPLETE

Value

data.frame with daily state variable

`epirice.define.param` *Define values of the parameters for the Epirice model*

Description

Define parameters values

Usage

```
epirice.define.param()
```

Value

matrix with parameter values (nominal, binf, bsup)

`epirice.model` *The Epirice model (Disease model for rice)*

Description

Model description. Adapted from Savary et al.(2012)

Usage

```
epirice.model(param, weather, sdate = 1, ldate = 120, H0 = 600)
```

Arguments

`param`	: a vector of parameters
`weather`	: weather data.frame for one single year
`sdate`	: date to begin simulation (day of year) (default 1)
`ldate`	: date to end simulation (day of year) (default 120)
`H0`	: initial number of plant's healthy sites(default 600)

Value

data.frame with daily state variable

See Also

`epirice.multi.simule`

epirice.multi.simule *Wrapper function to run Epirice multiple times (for multiple sets of inputs)*

Description

Wrapper function for epirice.model

Usage

```
epirice.multi.simule(param, multi.simule, all = FALSE)
```

Arguments

param	: a vector of parameters
multi.simule	: matrix of n row definition of input variable : site, year and date of transplantation.
all	: if you want a matrix combining multi.simule and output (default = FALSE)

Value

matrix with AUDPC for each input vector

See Also

epirice.model

epirice.weather *Read weather data for Epirice (southern Asia weather)*

Description

Read weather data and format them for epirice.model

Usage

```
epirice.weather(working.year = NA, working.site = NA,
  weather = weather_SouthAsia)
```

Arguments

working.year	: year for the subset of weather data (default=NA : all the year)
working.site	: site for the subset of weather data (default=NA : all the site)
weather	: weather table

Value

data.frame with daily weather data for one or several site(s) and for one or several year(s)

---

evaluation.criteria *Calcule multiple goodness-of-fit criteria*

Description

This function is depreciated and will be remove from the package in future versions. Please use goodness.of.fit

Usage

```
evaluation.criteria(Ypred, Yobs, draw.plot = FALSE)
```

Arguments

Ypred	: prediction values from the model
Yobs	: observed values
draw.plot	: draw evaluation plot

Value

data.frame with the different evaluation criteria

Examples

```
# observed and simulated values
obs<-c(78,110,92,75,110,108,113,155,150)
sim<-c(126,126,126,105,105,105,147,147,147)
evaluation.criteria(sim,obs,draw.plot=TRUE)
```

---

exponential.model *The Exponential growth model of dynamic of population*

Description

Exponential growth model of dynamic of population

Usage

```
exponential.model(a, Y0, duration = 40, dt = 1)
```

Arguments

a	: growth rate
Y0	: initial condition
duration	: duration of simulation
dt	: time step for integration

Value

data.frame with Y for each time step

See Also

verhulst.update for the update function of the Verhulst model.

---

exponential.model.bis *The Exponential growth model of dynamic of population - another form*

Description

Exponential growth model of dynamic of population - another form

Usage

```
exponential.model.bis(a, Y0, duration = 40, dt = 1)
```

Arguments

a	: growth rate
Y0	: initial condition
duration	: duration of simulation
dt	: time step for integration

Value

data.frame with Y for each time step

See Also

verhulst.update for the update function of the Verhulst model.

exponential.model.ie *The Exponential growth model of dynamic of population - with improved Euler integration*

Description

Exponential growth model of dynamic of population - with improved Euler integration

Usage

```
exponential.model.ie(a, Y0, duration = 40, dt = 1)
```

Arguments

a	: growth rate
Y0	: initial condition
duration	: duration of simulation
dt	: time step for integration

Value

data.frame with Y for each time step

See Also

verhulst.update for the update function of the Verhulst model.

goodness.of.fit *Calcule multiple goodness-of-fit criteria*

Description

Calcule multiple goodness-of-fit criteria

Usage

```
goodness.of.fit(Yobs, Ypred, draw.plot = FALSE)
```

Arguments

Yobs	: observed values
Ypred	: prediction values from the model
draw.plot	: draw evaluation plot

Value

data.frame with the different evaluation criteria

Examples

```
# observed and simulated values
obs<-c(78,110,92,75,110,108,113,155,150)
sim<-c(126,126,126,105,105,105,147,147,147)
goodness.of.fit(obs,sim,draw.plot=TRUE)
```

---

graph_epid *Plot output of a Classical SEIR model for plant disease*

Description

Plot the output of the Zadoks classical SEIR model for plant disease.

Usage

```
graph_epid(out, typel = "s", all = TRUE, param = TRUE)
```

Arguments

out	: output of the zadoks.original.model
typel	: type of plot (default : s)
all	: if all=true (default), plot all the state variable
param	: if param (default), add the values of param on the plot

Value

plot

See Also

zakoks.original.model

graph_epid_s *Plot output of a Classical SEIR model for plant disease*

Description

Plot the output of the Zadoks classical SEIR model for plant disease.

Usage

```
graph_epid_s(out, typel = "s", all = TRUE, param = TRUE)
```

Arguments

out	: output of the zadoks.original.model
typel	: type of plot (default : s)
all	: if all=true (default), plot all the state variable
param	: if param (default), add the values of param on the plot

Value

plot

See Also

zakoks.original.model

lactation.calf.model *The Lactation model*

Description

Model description. This model is a model of lactating mammary glands of cattle described by Heather et al. (1983). This model was then inspired more complex models based on these principles. This model simulates the dynamics of the production of cow's milk. the system is represented by 6 state variables: change in hormone levels (H), the production and loss of milk secreting cells (CS), and removing the secretion of milk (M), the average quantity of milk contained in the animal (Mmean), the amount of milk removed (RM) and yield (Y). The model has a time step dt = 0.1 for regular consumption of milk by a calf. The model is defined by a few equations, with a total of fourteen parameters for the described process.

Usage

```
lactation.calf.model(cu, kdiv, kdl, kdh, km, ksl, kr, ks, ksm, mh, mm, p,
  mum, rc, duration, dt)
```

Arguments

cu	: number of undifferentiated cells
kdiv	: cell division rate, Michaelis-Menten constant
kdl	: constant degradation of milk
kdh	: rate of decomposition of the hormone
km	: constant secretion of milk
ksl	: milk secretion rate, Michaelis-Menten constant
kr	: average milk constant
ks	: rate of degradation of the basal cells
ksm	: constant rate of degradation of milk secreting cells
mh	: parameter
mm	: storage Capacity milk the animal
p	: parameter
mum	: setting the maximum rate of cell division
rc	: parameter of milk m (t) function
duration	: duration of simulation
dt	: time step

Value

data.frame with CS, M, Mmoy, RM, day, week

Examples

```
lactation.calf.model2(lactation.define.param()["nominal",],300,0.1)
```

---

lactation.calf.model2 *The Lactation model for use with lactation.calf.simule*

Description

see lactation.calf.model for model description.

Usage

```
lactation.calf.model2(param, duration, dt)
```

Arguments

param	: a vector of parameters
duration	: duration of simulation
dt	: time step

Value

data.frame with CS, M, Mmoy, RM, day, week

Examples

```
sim=lactation.calf.model2(lactation.define.param()["nominal",],6+2*7, 0.1)
```

---

lactation.calf.simule *Wrapper function to run the Lactation model for multiple sets of parameter values*

Description

Wrapper function to run the Lactation model for multiple sets of parameter values

Usage

```
lactation.calf.simule(X, duration, dt)
```

Arguments

X	: parameter matrix
duration	: duration of simulation
dt	: time step

Value

data.frame with : number of paramter vector (line number from X), week, CS, M, Mmoy, RM, day, week

---

lactation.define.param *Define values of the parameters for the Lactation model*

Description

values from Heather et al. (1983) for different scenarios

Usage

```
lactation.define.param(type = "calf")
```

Arguments

type	: for which model version ? "calf" or "machine"

Value

matrix with parameter values (nominal, binf, bsup)

Examples

```
lactation.define.param()
```

`lactation.machine.model`

The Lactation model with milking machine

Description

Model description. This model is a model of lactating mammary glands of cattle described by Heather et al. (1983). This model was then inspired more complex models based on these principles. This model simulates the dynamics of the production of cow's milk. the system is represented by 6 state variables: change in hormone levels (H), the production and loss of milk secreting cells (CS), and removing the secretion of milk (M), the average quantity of milk contained in the animal (Mmean), the amount of milk removed (RM) and yield (Y). The model has a time step dt = 0.001 for milking machines. The model is defined by a few equations, with a total of twenty parameters for the described process.

Usage

```
lactation.machine.model(cu, kdiv, kdl, kdh, km, ksl, kr, ks, ksm, mh, mm,
  p, mum, rma, t1, t2, t3, t4, t5, t6, duration, dt, CSi, Mi)
```

Arguments

`cu`	: number of undifferentiated cells
`kdiv`	: cell division rate, Michaelis-Menten constant
`kdl`	: constant degradation of milk
`kdh`	: rate of decomposition of the hormone
`km`	: constant secretion of milk
`ksl`	: milk secretion rate, Michaelis-Menten constant
`kr`	: average milk constant
`ks`	: rate of degradation of the basal cells
`ksm`	: constant rate of degradation of milk secreting cells
`mh`	: parameter
`mm`	: storage Capacity milk the animal
`p`	: parameter
`mum`	: setting the maximum rate of cell division
`rma`	: parameter of milk m (t) function

t1	: parameter of milk m (t) function
t2	: parameter of milk m (t) function
t3	: parameter of milk m (t) function
t4	: parameter of milk m (t) function
t5	: parameter of milk m (t) function
t6	: parameter of milk m (t) function
duration	: duration of simulation
dt	: time step
CSi	: initial Number of secretory cells
Mi	: initial Quantity of milk in animal (kg)

Value

matrix with CS,M,Mmoy,RM

`lactation.machine.model2`

The Lactation model for use with lactation.machine.simule

Description

see lactation.calf.model for model description.

Usage

```
lactation.machine.model2(param, duration, dt, CSi, Mi)
```

Arguments

param	: a vector of parameters containning (cu,kdiv,kdl,kdh,km,ksl,kr,ks,ksm,mh,mm,p,mum,rma,t1,t2,t3,t4,t5,t lactation.model.machine)
duration	: duration of simulation
dt	: time step
CSi	: initial Number of secretory cells
Mi	: initial Quantity of milk in animal (kg)

Value

data.frame with CS, M, Mmoy, RM, day, week

magarey.define.param *Define values of the parameters for the Magarey model*

Description

Define values of the parameters for the Magarey model

Usage

```
magarey.define.param(species = "unkown")
```

Arguments

species	: name of a species. By default, value for an "unkown" species are given. Other possibility are "G.citricarpa" or "Kaki.fungus"

Value

matrix with parameter values (nominal, binf, bsup)

Examples

```
magarey.define.param(species="G.citricarpa")
magarey.define.param(species="Kaki.fungus")
```

magarey.model *The Magarey model*

Description

Generic model of infection for foliar diseases caused by fungi (from Magarey et al.,2005).

Usage

```
magarey.model(T, Tmin, Topt, Tmax, Wmin, Wmax)
```

Arguments

T	: input variable. Either a scalar or a vector (for a weather series).
Tmin	: parameter of minimal temperature for infection (degC)
Topt	: parameter of optimal temperature for infection (degC)
Tmax	: parameter of maximal temperature for infection (degC)
Wmin	: parameter of minimal wetness duration for infection (hour)
Wmax	: parameter of maximal wetness duration for infection (hour)

Value

Wetness duration (W, hour). Either a scalar or a vector depending on T.

Examples

```
plot(1:35, magarey.model (1:35,7, 18, 30, 10, 42), type="l", xlab="T", ylab="W")
```

---

magarey.model2 *The Magarey model, taking a vector of parameters as argument*

Description

Generic model of infection for foliar diseases caused by fungi (from Magarey et al.,2005).

Usage

```
magarey.model2(T, param)
```

Arguments

T	: input variable. Either a scalar or a vector (for a weather series).
param	: parameters

Value

W : Wetness duration (hour). Either a scalar or a vector depending on T.

---

magarey.simule *Wrapper function to run the Magarey model multiple times (for multiple sets of inputs)*

Description

Wrapper function to run the Magarey model multiple times (for multiple sets of inputs)

Example magarey.simule(magarey.define.param(),15)

Usage

```
magarey.simule(X, T, all = FALSE)
```

Arguments

X	: parameter matrix
T	: input variable, temperature
all	: if you want a matrix combining X and output

Value

a table with wetness duration (W) for each parameter vector

`maize.data_EuropeEU` *maize biomass and leaf area data*

Description

"observation data" for several site and year (site-year) in Europe. This data are fake observed date, derived from simulations with an error model.

Usage

```
maize.data_EuropeEU
```

Format

a `RangedData` instance, 1 row per measurement.

Source

NA

`maize.data_MetaModelling`
dataset of simulation for maize final biomass

Description

Simulation data for several site and year (site-year) in Europe. This data are from run of the original maize crop model with the R function maize.model for the 30 French sites and 17 years included in the dataset weather_FranceWest of the package ZeBook. Our training dataset includes 510 (30 sites * 17 years) simulated biomass values and the 510 corresponding series of input values. The input values are the three average temperatures T1, T2, T3 and the three average radiations RAD1, RAD2, RAD3 computed on 3 periods of the growing season. Period 1 : from day 1 to dat 50 (day of the year), period 2: from day 51 to day 100, period 3 : from day 101 to day 150.

Usage

```
maize.data_EuropeEU
```

Format

a `RangedData` instance, 1 row per simulation. Site, Year, T1, T2, T3, RAD1, RAD2, RAD3, B

Source

NA

See Also

maize.model, weather_FranceWest

`maize.define.param` *Define values of the parameters for the Maize model*

Description

Define parameters values

Usage

```
maize.define.param()
```

Value

matrix with parameter values (nominal, binf, bsup)

`maize.model` *The basic Maize model.*

Description

Model description. This model is a dynamic model of crop growth for Maize cultivated in potential conditions. The crop growth is represented by three state variables, leaf area per unit ground area (leaf area index, LAI), total biomass (B) and cumulative thermal time since plant emergence (TT). It is based on key concepts included in most crop models, at least for the "potential production" part. In fact, this model does not take into account any effects of soil water, nutrients, pests, or diseases,...

Usage

```
maize.model(Tbase, RUE, K, alpha, LAImax, TTM, TTL, weather, sdate, ldate)
```

Arguments

Tbase	: parameter the baseline temperature for growth (degreeCelsius)
RUE	: parameter radiation use efficiency (?)
K	: parameter extinction coefficient (relation between leaf area index and intercepted radiation) (-)
alpha	: parameter the relative rate of leaf area index increase for small values of leaf area index (?)
LAImax	: parameter maximum leaf area index (-)
TTM	: parameter temperature sum for crop maturity (degreeC.day)
TTL	: parameter temperature sum at the end of leaf area increase (degreeC.day)
weather	: weather data.frame for one single year
sdate	: sowing date
ldate	: last date

Details

The tree state variables are dynamic variables depending on days after emergence: TT(day), B(day), and LAI(day). The model has a time step dt of one day.
The model is defined by a few equations, with a total of seven parameters for the described process.
(1) $TT(day+1) = TT(day) + dTT(day)$
(2) $B(day+1) = B(day) + dB(day)$
(3) $LAI(day+1) = LAI(day) + dLAI(day)$
(4) $dTT(day) = \max(\frac{TMIN(day)+TMAX(day)}{2} - Tbase; 0)$
(5) $dB(day) = RUE * (1 - e^{-K*LAI(day)*I(day)}),\ if\ TT(day) \leq TTM$
$dB(day) = 0,\ if\ TT(day) > TTM$
(6) $dLAI(day) = alpha*dTT(day)*LAI(day)*\max(LAImax - LAI(day); 0),\ if\ TT(day) \leq TTL$
$dLAI(day) = 0,\ if\ TT(day) > TTL$

Value

data.frame with daily TT, LAI,B

See Also

maize.model2, maize.define.param, maize.simule, maize.multisy, maize.simule240,maize.simule_multisy240

Examples

```
weather = maize.weather(working.year=2010, working.site=30,weather_all=weather_EuropeEU)
maize.model(Tbase=7, RUE=1.85, K=0.7, alpha=0.00243, LAImax=7, TTM=1200, TTL=700,
  weather, sdate=100, ldate=250)
```

maize.model2 *The basic Maize model for use with maize.simule*

Description

Wrapper pour maize.model

Usage

```
maize.model2(param, weather, sdate, ldate)
```

Arguments

param	: a vector of parameters
weather	: weather data.frame for one single year
sdate	: sowing date
ldate	: last date

Value

data.frame with daily TT, LAI,B

Examples

```
weather = maize.weather(working.year=2010, working.site=30,weather_all=weather_EuropeEU)
maize.model2(maize.define.param()["nominal",], weather, sdate=100, ldate=250)
```

maize.muchow.graph *Plot dynamic output of Muchow Maize model.*

Description

Plot 6 graphs of main output variables of the Muchow Maize model.

Usage

```
maize.muchow.graph(res)
```

Arguments

res	: list of result from maize.muchow.model

See Also

mm.A.fct, mm.LN.fct, mm.FAS.fct, maize.multisy, mm.HI.fct,maize.muchow.model

Examples

```
# not run in package test
# res = maize.muchow.model(weather=maize.weather(working.year=2010, working.site=1))
# maize.muchow.graph(res)
```

`maize.muchow.model` *Maize model, with harvest index and yield from Muchow et al. (1990)*

Description

Maize model, with harvest index and yield. from Muchow RC, Sinclair TR, and Bennett JM (1990). Temperature and Solar Radiation Effects on Potential Maize Yield across Locations AGRONOMY JOURNAL, VOL. 82, MARCH-APRIL 1990

Usage

```
maize.muchow.model(Tbase1 = 8, TTE = 87, TTS = 67, Tbase2 = 0,
  TTRUE = 500, TTM = 1150, TLN = 20, AM = 596, RUE1 = 1.6,
  RUE2 = 1.2, K = 0.4, HImax = 0.5, Population = 7, sdate = 100,
  ldate = 365, weather)
```

Arguments

`Tbase1`	: base temperature before silking(degC)
`TTE`	: Thermal units from sowing to emergence/leaf growth (degC.day)
`TTS`	: Thermal units from end of leaf growth to silking (degC.day)
`Tbase2`	: base temperature after silking (degC)
`TTRUE`	: Thermal units from silking for RUE change (degC.day)
`TTM`	: Thermal units from silking to physiological maturity (degC.day)
`TLN`	: total number of leaves initiated (-)
`AM`	: area of the largest leaf (cm2)
`RUE1`	: radiation use efficiency (g.MJ-1) from crop emergence until 500 thermal units (base 0 "C) after silking
`RUE2`	: radiation use efficiency (g.MJ-1) from 500 thermal units (base 0 "C) after silking
`K`	: radiation extinction coefficient (-)
`HImax`	: maximum harvest index - genetic potential (-)
`Population`	: number of plant per square meter (-)
`sdate`	: sowing date (day)
`ldate`	: end of simulation (day)
`weather`	: daily weather dataframe

Value

data.frame with TT1, TT2, STADE, LN, LAI, B, HI, YIELD

See Also

mm.A.fct, mm.LN.fct, mm.FAS.fct, maize.multisy, mm.HI.fct,maize.muchow.graph

Examples

```
# not run in package test
# res = maize.muchow.model(weather=maize.weather(working.year=2010, working.site=1))
#res$FinalYield
```

`maize.multisy` *Wrapping function to run maize model on several site-years*

Description

Wrapping function to run maize model on several site-years

Usage

```
maize.multisy(param, list_site_year, sdate, ldate,
  weather_all = weather_EuropeEU)
```

Arguments

`param`	: a vector of parameters
`list_site_year`	: vector of site-year
`sdate`	: sowing date
`ldate`	: last date
`weather_all`	: weather data.frame for corresponding site-years

Value

a data.frame with simulation for all site-years, with the first column sy indicating the site-years

`maize.multisy240` *Wrapper function to run Maize model for multiple sets of input variables (site-year) and give Biomass at day240.*

Description

Wrapper function to run Maize model for multiple sets of input variables (site-year) and give Biomass at day240.

Usage

```
maize.multisy240(param, liste_sy, sdate, ldate,
  weather_all = weather_EuropeEU)
```

Arguments

`param`	: a vector of parameters
`liste_sy`	: vector of site-year
`sdate`	: sowing date
`ldate`	: last date
`weather_all`	: weather data table used

Value

mean biomass at day=240

Examples

```
maize.multisy240(maize.define.param()["nominal",],c("18-2006","64-2004") , sdate=100, ldate=250)
```

`maize.RUEtemp` *Calculate effect of temperature on RUE for Maize*

Description

Function to compute effect of temperature on RUE

Usage

```
maize.RUEtemp(T, RUE_max, T0, T1, T2, T3)
```

Arguments

T	: temperature
RUE_max	: maximum value for RUE
T0	: temperature parameter
T1	: temperature parameter
T2	: temperature parameter
T3	: temperature parameter

Value

RUE value

---

maize.simule *Wrapper function to run Maize model for multiple sets of parameter values*

Description

wrapper for maize.model2

Usage

```
maize.simule(X, weather, sdate, ldate, all = FALSE)
```

Arguments

X	: matrix of n row vectors of 7 parameters
weather	: weather data.frame for one single year
sdate	: sowing date
ldate	: last date
all	: if you want a matrix combining X and output (default = FALSE)

Value

matrix with maximum biomass for each parameter vector

`maize.simule240` *Wrapper function to run Maize model multiple times for multiple sets of parameter values and give Biomass at day240*

Description

Wrapper function for multiple simulation with Maize model

Usage

```
maize.simule240(X, weather, sdate, ldate, all = FALSE)
```

Arguments

`X`	: matrix of n row vectors of 7 parameters
`weather`	: weather data.frame for one single year
`sdate`	: sowing date
`ldate`	: last date
`all`	: if you want a matrix combining X and output (default = FALSE)

Value

a matrix of biomass at day=240 for all combinations of parameters of X

Examples

```
sy="18-2006"
weather = maize.weather(working.year=strsplit(sy,"-")[[1]][2],
  working.site=strsplit(sy,"-")[[1]][1],weather_all=weather_EuropeEU)
maize.simule240(maize.define.param(),weather, sdate=100, ldate=250, all=FALSE)
```

`maize.simule_multisy240` *Wrapper function to run Maize model for multiple sets of parameter values (virtual design) and multiple sets of input variables (site-year) and give Biomass at day240*

Description

Wrapper function to run Maize model for multiple sets of input variables (site-year) and give Biomass at day240.

Usage

```
maize.simule_multisy240(X, liste_sy, sdate, ldate, all = FALSE)
```

Arguments

X	: matrix of n row vectors of 7 parameters
liste_sy	: vector of site-year
sdate	: sowing date
ldate	: last date
all	: if you want a matrix combining X and output (default = FALSE)

Value

a matrix of mean biomass at day=240 for all combinations of parameters of X

Examples

```
maize.simule_multisy240(maize.define.param(),c("18-2006","64-2004"),
  sdate=100, ldate=250, all=FALSE)
```

---

maize.weather *Read weather data for the Maize model*

Description

Function to read weather data and format them for maize.model

Usage

```
maize.weather(working.year = NA, working.site = NA,
  weather_all = weather_FranceWest)
```

Arguments

working.year	: year for the subset of weather data (default=NA : all the year)
working.site	: site for the subset of weather data (default=NA : all the site)
weather_all	: weather data base (default=weather_FranceWest)

Value

data.frame with daily weather data for one or several site(s) and for one or several year(s)

maize_cir.model *The Maize model with additional state variable CumInt*

Description

Variant of the maize model

Usage

```
maize_cir.model(Tbase, RUE, K, alpha, LAImax, TTM, TTL, weather, sdate,
  ldate)
```

Arguments

Tbase	: parameter the baseline temperature for growth (degreeCelsius)
RUE	: parameter radiation use efficiency (?)
K	: parameter extinction coefficient (relation between leaf area index and intercepted radiation) (-)
alpha	: parameter the relative rate of leaf area index increase for small values of leaf area index (?)
LAImax	: parameter maximum leaf area index (-)
TTM	: parameter temperature sum for crop maturity (degreeC.day)
TTL	: parameter temperature sum at the end of leaf area increase (degreeC.day)
weather	: weather data.frame for one single year
sdate	: sowing date
ldate	: last date

Value

data.frame with daily TT, LAI,B

maize_cir_rue.model *The Maize model with temperature dependent RUE and CumInt*

Description

Variant of the maize.model

Usage

```
maize_cir_rue.model(Tbase, RUE_max, K, alpha, LAImax, TTM, TTL, weather,
  sdate, ldate)
```

Arguments

Tbase	: parameter the baseline temperature for growth (degreeCelsius)
RUE_max	: parameter maximum radiation use efficiency (?)
K	: parameter extinction coefficient (relation between leaf area index and intercepted radiation) (-)
alpha	: parameter the relative rate of leaf area index increase for small values of leaf area index (?)
LAImax	: parameter maximum leaf area index (-)
TTM	: parameter temperature sum for crop maturity (degreeC.day)
TTL	: parameter temperature sum at the end of leaf area increase (degreeC.day)
weather	: weather data.frame for one single year
sdate	: sowing date
ldate	: last date

Value

data.frame with daily TT, LAI,B

---

maize_cir_rue_ear.model

The Maize model with temperature dependent RUE, CumInt and ear growth

---

Description

Variant of the maize.model

Usage

```
maize_cir_rue_ear.model(Tbase, RUE_max, K, alpha, LAImax, TTM, TTL,
  weather, sdate, ldate)
```

Arguments

Tbase	: parameter the baseline temperature for growth (degreeCelsius)
RUE_max	: parameter maximum radiation use efficiency (?)
K	: parameter extinction coefficient (relation between leaf area index and intercepted radiation) (-)
alpha	: parameter the relative rate of leaf area index increase for small values of leaf area index (?)
LAImax	: parameter maximum leaf area index (-)
TTM	: parameter temperature sum for crop maturity (degreeC.day)

TTL	: parameter temperature sum at the end of leaf area increase (degreeC.day)
weather	: weather data.frame for one single year
sdate	: sowing date
ldate	: last date

Value

data.frame with daily TT, LAI,B

`mm.A.fct` *Expanded leaf area function for Muchow et al. (1990) Maize model*

Description

Compute fully expanded area by leaf number (A, cm2)

Usage

```
mm.A.fct(LN, AM, LNM, a1 = -0.0344, a2 = 0.000731)
```

Arguments

LN	: Leaf number
AM	: area of the largest leaf (cm2)
LNM	: leaf number having the largest area (-)
a1	: coefficient of the statistical relation (default : -0.0344)
a2	: coefficient of the statistical relation (default : 0.000731)

Value

vector of Expanded leaf area

See Also

maize.muchow.model, mm.LN.fct, mm.FAS.fct, maize.multisy, mm.HI.fct,maize.muchow.graph

Examples

```
barplot(mm.A.fct(LN=1:20, AM=750, LNM=12),names.arg=1:20,
horiz=TRUE,xlab="leaf area (cm2)",ylab="leaf number")
```

mm.FAS.fct *Senescence function for Muchow et al. (1990) Maize model*

Description

Senesced fraction of total leaf area (FAS) increase with thermal units (TU) from emergence

Usage

```
mm.FAS.fct(TT, TTE, c1 = 0.00161, c2 = 0.00328)
```

Arguments

TT	: Thermal time (degC.day)
TTE	: Thermal units from sowing to emergence/leaf growth (degC.day)
c1	: coefficient of the statistical relation (default : 0.00161)
c2	: coefficient of the statistical relation (default : 0.00328)

Value

Senesced fraction of total leaf area

See Also

maize.muchow.model, mm.A.fct, mm.LN.fct, maize.multisy, mm.HI.fct,maize.muchow.graph

Examples

```
plot(1:2500,mm.FAS.fct(1:2500,TTE=87))
```

mm.HI.fct *Harvest index function for Muchow et al. (1990) Maize model*

Description

Compute the harvest index.

Usage

```
mm.HI.fct(day, daysilking, HImax, d1 = 0.015, d2 = 3)
```

Arguments

day	: day of the year
daysilking	: day of the year for silking (day)
HImax	: maximum harvest index - genetic potential (-)
d1	: coefficient of the statistical relation (day-1, default : 0.015)
d2	: coefficient of the statistical relation (day, default : 3)

Value

Harvest index

See Also

maize.muchow.model, mm.A.fct, mm.LN.fct, maize.multisy, mm.FAS.fct,maize.muchow.graph

Examples

```
plot(1:350, mm.HI.fct(1:350, 200, 0.75), type="l")
```

---

mm.LN.fct *Leaf number function for Muchow et al. (1990) Maize model*

Description

Leaf number as a function of thermal time

Usage

```
mm.LN.fct(TT1, TTE, b1 = 2.5, b2 = 0.00225, TLN = 20)
```

Arguments

TT1	: Thermal time from sowing (degC.day)
TTE	: Thermal units from sowing to emergence/leaf growth (degC.day)
b1	: coefficient of the statistical relation (default : 2.5)
b2	: coefficient of the statistical relation (default : 0.00225)
TLN	: total number of leaves initiated (-)

Value

Leaf number

See Also

maize.muchow.model, mm.A.fct, mm.FAS.fct, maize.multisy, mm.HI.fct,maize.muchow.graph

Examples

```
plot(1:1000,mm.LN.fct(1:1000,TTE=87))
```

`param.rtriangle` *Generate a random plan as a data frame. Columns are parameters. Values have triangle distribution*

Description

according to nominal, minimal and maximal values defined in a model.factors matrix

Usage

```
param.rtriangle(model.factors, N)
```

Arguments

`model.factors`	: matrix defining nominal, minimal (binf), maximal values (bsup) for a set of p parameters
`N`	: size of sample

Value

parameter matrix of dim = (N, p)

`param.runif` *Generate a random plan as a data frame. Columns are parameters. Values have uniform distribution*

Description

according to minimal and maximal values defined in a model.factors matrix

Usage

```
param.runif(model.factors, N)
```

Arguments

`model.factors`	: matrix defining minimal (binf) and maximal values (bsup) for a set of p parameters
`N`	: size of sample

Value

parameter matrix of dim = (N, p)

`population.age.matrix.model` *The PopulationAge model (Population Dynamics with Age Classes) - matrix form*

Description

Population Dynamics Model with Age Classes for an insect Exactly the same model as population.age.model, but written as a matrix computation. It's possible for this model. It's really more efficient and reduce computer time by 6! 7 states variables E : egg stage. homogenous population (density) (number per ha) L1 : larvae1 stage. homogenous population (density) (number per ha) L2 : larvae2 stage. homogenous population (density) (number per ha) L3 : larvae3 stage. homogenous population (density) (number per ha) L4 : larvae4 stage. homogenous population (density) (number per ha) P : pupae stage. homogenous population (density) (number per ha) A : adult stage. homogenous population (density) (number per ha)

Usage

```
population.age.matrix.model(rb = 3.5, mE = 0.017, rE = 0.172,
  m1 = 0.06, r12 = 0.217, m2 = 0.032, r23 = 0.313, m3 = 0.022,
  r34 = 0.222, m4 = 0.02, r4P = 0.135, mP = 0.02, rPA = 0.099,
  mA = 0.027, iA = 0, duration = 100, dt = 1)
```

Arguments

`rb`	: eggs laid per adult per unit area (day-1)
`mE`	: relative mortality rate of egg (day-1)
`rE`	: eggs hatch (day-1)
`m1`	: relative mortality rate of larvae L1 (day-1)
`r12`	: relative rate L1->L2 (day-1)
`m2`	: relative mortality rate of larvae L2 (day-1)
`r23`	: relative rate L2->L3 (day-1)
`m3`	: relative mortality rate of larvae L3 (day-1)
`r34`	: relative rate L3->L4 (day-1)
`m4`	: relative mortality rate of larvae L4 (day-1)
`r4P`	: relative rate L4->P (day-1)
`mP`	: relative mortality rate of purpae (day-1)
`rPA`	: relative rate P->A (day-1)
`mA`	: relative mortality rate of adult L1 (day-1)
`iA`	: input rate of adult (unit.day-1)
`duration`	: simulation duration
`dt`	: time step for integration

Value

data.frame with values for state variables for each time step.

`population.age.model` *The PopulationAge model (Population Dynamics with Age Classes)*

Description

Population Dynamics Model with Age Classes for an insect

Usage

```
population.age.model(rb = 3.5, mE = 0.017, rE = 0.172, m1 = 0.06,
  r12 = 0.217, m2 = 0.032, r23 = 0.313, m3 = 0.022, r34 = 0.222,
  m4 = 0.02, r4P = 0.135, mP = 0.02, rPA = 0.099, mA = 0.027,
  iA = 0, duration = 100, dt = 1)
```

Arguments

`rb`	: eggs laid per adult per unit area (day-1)
`mE`	: relative mortality rate of egg (day-1)
`rE`	: eggs hatch (day-1)
`m1`	: relative mortality rate of larvae L1 (day-1)
`r12`	: relative rate L1->L2 (day-1)
`m2`	: relative mortality rate of larvae L2 (day-1)
`r23`	: relative rate L2->L3 (day-1)
`m3`	: relative mortality rate of larvae L3 (day-1)
`r34`	: relative rate L3->L4 (day-1)
`m4`	: relative mortality rate of larvae L4 (day-1)
`r4P`	: relative rate L4->P (day-1)
`mP`	: relative mortality rate of purpae (day-1)
`rPA`	: relative rate P->A (day-1)
`mA`	: relative mortality rate of adult L1 (day-1)
`iA`	: input rate of adult (unit.day-1)
`duration`	: simulation duration
`dt`	: time step for integration

Value

data.frame with values for state variables for each time step.

population.age.model.ode *The PopulationAge model (Population Dynamics with Age Classes) - ode form*

Description

Population Dynamics Model with Age Classes for an insect Exactly the same model as population.age.model, but written as an ordinary differential equation system (ode) with deSolve package. 7 states variables E : egg stage. homogenous population (density) (number per ha) L1 : larvae1 stage. homogenous population (density) (number per ha) L2 : larvae2 stage. homogenous population (density) (number per ha) L3 : larvae3 stage. homogenous population (density) (number per ha) L4 : larvae4 stage. homogenous population (density) (number per ha) P : pupae stage. homogenous population (density) (number per ha) A : adult stage. homogenous population (density) (number per ha)

Usage

```
population.age.model.ode(rb = 3.5, mE = 0.017, rE = 0.172,
  m1 = 0.06, r12 = 0.217, m2 = 0.032, r23 = 0.313, m3 = 0.022,
  r34 = 0.222, m4 = 0.02, r4P = 0.135, mP = 0.02, rPA = 0.099,
  mA = 0.027, iA = 0, duration = 100, dt = 1, method = "euler")
```

Arguments

rb	: eggs laid per adult per unit area (day-1)
mE	: relative mortality rate of egg (day-1)
rE	: eggs hatch (day-1)
m1	: relative mortality rate of larvae L1 (day-1)
r12	: relative rate L1->L2 (day-1)
m2	: relative mortality rate of larvae L2 (day-1)
r23	: relative rate L2->L3 (day-1)
m3	: relative mortality rate of larvae L3 (day-1)
r34	: relative rate L3->L4 (day-1)
m4	: relative mortality rate of larvae L4 (day-1)
r4P	: relative rate L4->P (day-1)
mP	: relative mortality rate of purpae (day-1)
rPA	: relative rate P->A (day-1)
mA	: relative mortality rate of adult L1 (day-1)
iA	: input rate of adult (unit.day-1)
duration	: simulation duration
dt	: time step for integration
method	: integration method (euler, rk4,...)

Value

data.frame with values for state variables for each time step.

`predator.prey.model` *The PredatorPrey model (Predator-Prey Lotka-Volterra with logistic equation for prey)*

Description

Predator-Prey Lotka-Volterra model (with logistic prey)

Usage

```
predator.prey.model(grH = 1, kH = 10, mrH = 0.2, eff = 0.5,
  mrA = 0.2, H0 = 1, A0 = 2, duration = 200, dt = 1,
  method = "euler")
```

Arguments

`grH`	: relative rate of prey population growth
`kH`	: environment carrying capacity for prey (number per ha)
`mrH`	: maximum predation rate (number per predator and per prey per day)
`eff`	: efficiency, growth of predator population depending on predation (-)
`mrA`	: mortality of predator (-)
`H0`	: size of population of prey, at time 0
`A0`	: size of population of predator, at time 0
`duration`	: simulation duration
`dt`	: time step for integration
`method`	: integration method

Value

data.frame with daily H and A

q.arg.fast.runif *Build the q.arg argument for the FAST function (sensitivity analysis)*

Description

according to minimal and maximal values defined in a model.factors matrix

Usage

```
q.arg.fast.runif(model.factors)
```

Arguments

model.factors	: matrix defining minimal (binf) and maximal values (bsup) for a set of p parameters

Value

a list of list

seedweight.data *Wheat grain weight measurements after anthesis*

Description

Darroch and Baker (1990) studied grain filling in three spring wheat genotypes. The data are seed weights of the spring wheat cultivar Neepawa in three different years 1986, 1987, 1988. These data were numerised from figure of the article, so they present slight difference with original data.

Usage

```
seedweight.data
```

Format

a RangedData instance, 1 row per measurement. DD = Degree Days after anthesis; seedweight = Wheat grain weight (mg)

Source

Darroch, B A, and R J Baker. 1990. "Grain filling in three spring wheat genotypes: statistical analysis." Crop Science 30(3):525-529.

seedweight.model *The SeedWeight model*

Description

The SeedWeight model is a logistic model of grain weight over time in wheat. The model was proposed by Darroch & Baker (1990) in a study of grain filling in three spring wheat genotypes. This model has a single input variable, degree days after anthesis noted DD, and three parameters, noted W, B and C. Parameters are estimated from observations.

Usage

```
seedweight.model(DD, W, B, C)
```

Arguments

DD	: degree days after anthesis
W	: parameter of the model
B	: parameter of the model
C	: parameter of the model

Value

Seed Weight for each TT

Examples

```
plot(1:500,seedweight.model(1:500, W=30,B=4,C=0.020),type="l",
   xlab="degree days after anthesis", ylab="grain weight")
```

Sunflower_Phomopsis *Phomopsis stem canker observations for Sunflower*

Description

This dataset contains fraction intercepted photosynthetically active radiation (IPAR) and corresponding percent of girdling lesions at harvest (pclesions) for 43 fields. Phomopsis stem canker is a worldwide fungal disease of sunflower, which causes stem girdling lesions and a consequent reduction in yield. One wants to decide if the number of girdling lesions at harvest in the absence of early treatment will exceed 15 This data frame consist of a sample of fields with values for IPAR (ipar) and for the percent of girdling lesions at harvest (pclesions).

Usage

```
Sunflower_Phomopsis
```

Format

a RangedData instance, 1 row per observation.

Source

Debaeke, P., & Estragnat, A. (2009). Crop canopy indicators for the early prediction of Phomopsis stem canker (Diaporthe helianthi) in sunflower. Crop Protection, 28(9), 792-801. doi:10.1016/j.cropro.2009.04.011

---

threshold.measures *Computation of threshold.measures*

Description

Computation of threshold.measures

Usage

```
threshold.measures(Yobs, Ypred, p, d, units = "")
```

Arguments

Yobs	: observed values
Ypred	: prediction values from the model
p	: TO COMPLETE
d	: TO COMPLETE
units	: units

Value

data.frame with the different evaluation criteria

Examples

```
# observed and simulated values
obs<-c(78,110,92,75,110,108,113,155,150)
sim<-c(126,126,126,105,105,105,147,147,147)
threshold.measures(obs,sim,80,1.0)
```

verhulst.model — *The Verhulst (logistic) model - calculate daily values over designated time period*

Description

The Verhulst (logistic) model - calculate daily values over designated time period

Usage

```
verhulst.model(a, k, Y0, duration)
```

Arguments

a	: growth rate
k	: capacity
Y0	: initial condition
duration	: duration of simulation

Value

data.frame with daily Y

See Also

verhulst.update for the update function of the Verhulst model.

Examples

```
plot(verhulst.model(0.08,100,1,100), type="l", ylim=c(0,115),
   xlab="day", ylab="Y, population density",lwd=2)
```

verhulst.update — *The Verhulst (logistic) model - calculate change for one day*

Description

The Verhulst (logistic) model - calculate change for one day

Usage

```
verhulst.update(Y, a, k)
```

Arguments

Y	: state variable Y(t=day)
a	: growth rate
k	: capacity

Value

state variable at Y(t=day+1)

See Also

verhulst.model for the integration loop function of the Verhulst model.

---

watbal.define.param *Define values of the parameters for the WaterBalance model*

Description

Define values of the parameters for the WaterBalance model

Usage

```
watbal.define.param()
```

Value

matrix with parameter values (nominal, binf, bsup)

---

watbal.model *WaterBalance model - calculate soil water over designated time period*

Description

WaterBalance model - calculate soil water over designated time period

Usage

```
watbal.model(param, weather, WP, FC, WAT0 = NA)
```

Arguments

param	: a vector of parameters
weather	: weather data.frame for one single year
WP	: Water content at wilting Point (cm^3.cm^-3)
FC	: Water content at field capacity (cm^3.cm^-3)
WAT0	: Initial Water content (mm). If NA WAT0=z*FC

Value

data.frame with daily RAIN, ETR, Water at the beginning of the day (absolute : WAT, mm and relative value : WATp, -)

`watbal.model.arid` *WaterBalance model - Variant with another order of calculation and ARID index*

Description

WaterBalance model - Variant with another order of calculation and ARID index

Usage

```
watbal.model.arid(WHC, MUF, DC, z, CN, weather, WP, FC, WAT0 = NA)
```

Arguments

`WHC`	: Water Holding Capacity of the soil (cm^3 cm^-3)
`MUF`	: Water Uptake coefficient (mm^3 mm^-3)
`DC`	: Drainage coefficient (mm^3 mm^-3)
`z`	: root zone depth (mm)
`CN`	: Runoff curve number
`weather`	: weather data.frame for one single year
`WP`	: Water content at wilting Point (cm^3.cm^-3)
`FC`	: Water content at field capacity (cm^3.cm^-3)
`WAT0`	: Initial Water content (mm). If NA WAT0=z*FC

Value

data.frame with daily RAIN, ETR, Water at the beginning of the day (absolute : WAT, mm and relative value : WATp, -)

watbal.simobsdata — *Soil water content measurements and associated simulations with WaterBalance model*

Description

Data of soil water content from Luc Champolivier (CETIOM), in En Crambade (31, France), on canola without irrigation in 2008. sonde Diviner 2000 (from Sentek Pty Ltd) Simulation are from watbal.model, with an initial water content estimated from measurement with Diviner 2000.

Usage

```
watbal.simobsdata
```

Format

a RangedData instance, 1 row per day : Weather : day / RAIN / ETr / simulation : WAT / WATp / ARID observation :t1_WATp_0_40cm / t2_WATp_0_40cm / t3_WATp_0_40cm / WATp_SF.mean / WATp_SF.var

Source

CETIOM, Luc Champolivier (2008).

watbal.update — *WaterBalance model - calculate change in soil water for one day*

Description

WaterBalance model - calculate change in soil water for one day

Usage

```
watbal.update(WAT0, RAIN, ETr, param, WP, FC)
```

Arguments

WAT0	: Water at the beginning of the day (mm).
RAIN	: Rainfall of day (mm)
ETr	: Evapotranspiration of day (mm)
param	: a vector of parameters
WP	: Water content at wilting Point (cm^3.cm^-3)
FC	: Water content at field capacity (cm^3.cm^-3)

Value

WAT1 : Water at the beginning of the day+1 (mm).

watbal.weather — *Read weather data for the WaterBalance model (West of France Weather)*

Description

Read weather data for the WaterBalance model (West of France Weather)

Usage

```
watbal.weather(working.year = NA, working.site = NA)
```

Arguments

working.year	: year for the subset of weather data (default=NA : all the year)
working.site	: site for the subset of weather data (default=NA : all the site)

Value

data.frame with daily weather data for one or several site(s) and for one or several year(s)

weather_EuropeEU — *Weather serie for Europe EU from NASA POWER agroclimatology*

Description

This contemporary daily climate dataset for Europe covers the period 1st January 2001 to 31 December 2010 with 10 complete years of data. It cover a part of Europe) with an elevation less than 500m. The dataset was was extrated from the NASA Langley Research Center POWER Project which provide agroclimatology dataset (Chandler et al., 2004). It was funded through the NASA Earth Science Directorate Applied Science Program This climate datasetcontains daily estimates of precipitation, mean, minimum and maximum temperature, relative humidity, dew point, solar radiation and wind speed with global coverage at one degree resolution (approximately 111 km at the equator). The NASA POWER agroclimatology data are derived from various sources: solar radiation from satellite observations, meteorological data from the Goddard Earth Observing System global assimilation model version 4 (GEOS-4), and precipitation from the Global Precipitation Climate Project and Topical Rainfall Measurement Mission. A full description can be found at https://power.larc.nasa.gov/common/php/POWER_AboutAgroclimatology.php Elevation (Altitude) were retrive from Aster Global Digital Elevation Model by using the Webservice api.geonames.org/astergdem? Sample are: ca 30m x 30m, between 83N and 65S latitude. Result : a single number giving the elevation in meters according to aster gdem, ocean areas have been masked as "no data" and have been assigned a value of -9999 Example http://api.geonames.org/astergdem?lat=50.01&lng=10.2&user

Usage

```
weather_EuropeEU
```

Format

a RangedData instance, 1 row per day. SRAD daily Insolation Incident On A Horizontal Surface (MJ/m^2/day) T2M Average Air Temperature At 2 m Above The Surface Of The Earth (degrees C) TMIN Minimum Air Temperature At 2 m Above The Surface Of The Earth (degrees C) TMAX Maximum Air Temperature At 2 m Above The Surface Of The Earth (degrees C) RH2M Relative Humidity At 2 m (TDEW Dew/Frost Point Temperature At 2 m (degrees C) RAIN Average Precipitation (mm/day) WIND Wind Speed At 10 m Above The Surface Of The Earth (m/s)

Source

http://power.larc.nasa.gov/ and http://asterweb.jpl.nasa.gov/gdem.asp and http://www.geonames.org/about.html

---

weather_FranceWest *Weather series for western France from NASA POWER agroclimatology*

Description

This contemporary daily climate dataset for West of France covers the period 1st January 1984 to 31 December 2011. The precipitation data is limited to the period Jan-1997 Aug-2009, thus only 12 complete years of data were available for analysis involving precipitation(1997 to 2009). It cover main part of West part of France defined as a rectangle. The dataset was extracted from the NASA Langley Research Center POWER Project which provide agroclimatology dataset (Chandler et al., 2004). It was funded through the NASA Earth Science Directorate Applied Science Program This climate dataset contains daily estimates of precipitation, mean, minimum and maximum temperature, relative humidity, dew point, solar radiation and wind speed with global coverage at one degree resolution (approximately 111 km at the equator). The NASA POWER agroclimatology data are derived from various sources: solar radiation from satellite observations, meteorological data from the Goddard Earth Observing System global assimilation model version 4 (GEOS-4), and precipitation from the Global Precipitation Climate Project and Topical Rainfall Measurement Mission. A full description can be found at https://power.larc.nasa.gov/common/php/POWER_AboutAgroclimatology.php

Usage

```
weather_FranceWest
```

Format

a RangedData instance, 1 row per day.

Source

http://power.larc.nasa.gov/

weather_GNS *Weather series for Gainesville (FL, USA) years 1982 and 1983*

Description

This daily climate dataset for Gainesville (FL, USA) years 1982 and 1983 covers the period 1st January 1982 to 31 December 1983 with 2 complete years of data with precipitation. The dataset was provide by JJW Jones and the university of Florida to run simulation of the publication of Muchow et al. (1990). This climate dataset contains daily estimates of precipitation (RAIN), minimum (Tmin) and maximum temperature (Tmax), solar radiation (I), photosynthically active radiation (PAR)

Usage

```
weather_GNS
```

Format

a `RangedData` instance, 1 row per day.

Source

University of Florida

weather_SouthAsia *Weather series for southern Asia from NASA POWER agroclimatology*

Description

This contemporary daily climate dataset for South Asia covers the period 1st January 1997 to 31 December 2008 with 12 complete years of data with precipitation. It cover a part of South Asia (North-East of India, Bangladesh, Myanmar, Neapal) with an elevation less than 2500m. The dataset was extracted from the NASA Langley Research Center POWER Project which provide agroclimatology dataset (Chandler et al., 2004). It was funded through the NASA Earth Science Directorate Applied Science Program This climate dataset contains daily estimates of precipitation, mean, minimum and maximum temperature, relative humidity, dew point, solar radiation and wind speed with global coverage at one degree resolution (approximately 111 km at the equator). The NASA POWER agroclimatology data are derived from various sources: solar radiation from satellite observations, meteorological data from the Goddard Earth Observing System global assimilation model version 4 (GEOS-4), and precipitation from the Global Precipitation Climate Project and Topical Rainfall Measurement Mission. A full description can be found at https://power.larc.nasa.gov/common/php/POWER_AboutAgroclimatology.php Elevation (Altitude) were retrive from Aster Global Digital Elevation Model by using the Webservice api.geonames.org/astergdem? Sample are: ca 30m x 30m, between 83N and 65S latitude. Result : a single number giving the elevation in meters according to aster gdem, ocean areas have been masked as "no data" and have been assigned a value of -9999 Example http://api.geonames.org/astergdem?lat=50.01&lng=10.2&user

Usage

```
weather_SouthAsia
```

Format

a RangedData instance, 1 row per day.

Source

http://power.larc.nasa.gov/ and http://asterweb.jpl.nasa.gov/gdem.asp and http://www.geonames.org/about.html

---

weed.define.param *Define parameter values of the Weed model*

Description

Define parameter values of the Weed model

Usage

```
weed.define.param()
```

Value

matrix with parameter values (nominal, binf, bsup)

---

weed.model *The Weed model - calculate daily values over designated time period*

Description

The Weed model - calculate daily values over designated time period

Usage

```
weed.model(param, weed.deci)
```

Arguments

param	: vector of the 16 parameters
weed.deci	: decisions table for Soil, Crop et Herbicide

Value

data.frame with annual values of yield

weed.simule — *Wrapper function to run the Weed model multiple times (for multiple sets of inputs)*

Description

Wrapper function to run the Weed model multiple times (for multiple sets of inputs)

Usage

```
weed.simule(X, weed.deci)
```

Arguments

X	: parameter matrix
weed.deci	: decisions table for Soil, Crop et Herbicide

Value

matrix with Yield for year 3 for each parameter vector

weed.update — *The Weed model - calculate change for one year*

Description

The Weed model - calculate change for one year

Usage

```
weed.update(d, S, SSBa, DSBa, Soil, Crop, Herb, param)
```

Arguments

d	: weed density at seed emergence (plants/m2) - value for year
S	: seed production per m2 - value for year
SSBa	: surface seedbank after tillage (grains/m2) - value for year
DSBa	: deep seedbank after tillage (grains/m2) - value for year
Soil	: value for soil decision (1 or 0)
Crop	: value for crop decision (1 or 0)
Herb	: value for herbicide treatment decision (1 or 0)
param	: vector of the 16 parameters

Value

a vector with values of state variables for year+1

WheatYieldGreece *National Wheat Yield evolution for Greece from FAO*

Description

Wheat yield time series data in Greece from 1961 to 2010 yield.

Usage

```
WheatYieldGreece
```

Format

a RangedData instance, 1 row per measurement. Year, Yield : Wheat Yield (hectogram/hectare = 0.0001 ton/hectare)

Source

Food And Agricultural Organization of United Nations (FAO), FAOSTAT database, http://faostat.fao.org

Wheat_GPC *Grain Protein Contents in Wheat Grains*

Description

This dataset contains data for 43 plots. The column GPC corresponds to measured data, GPC.model1 and GPC.model2 correspond to the results obtained by two models. The other columns contain other information for each plot (technical practices) : number, tillage, max_water, preceding_crop, sow_date, Nendwinter, Nfertilizer, SPAD, NNI, Yield.

Usage

```
Wheat_GPC
```

Format

a RangedData instance, 1 row per plot.

Source

Barbottin et al. 2008

`zakoks.original.model` *Classical SEIR model for plant diseases from Zadoks (1971)*

Description

Model description. This model is a classical SEIR model for plant disease. It was written from it description included in the original publication of Zadoks (1971)

Usage

```
zakoks.original.model(nlpd = 4 * 10, nipd = 1 * 10, dmfr = 16,
  SITE0 = 5 * 10^9, weather, sdate = 145, ldate = 145 + 50,
  XLAT0 = 1)
```

Arguments

`nlpd`	: latent period (in degree.day) - default value for Puccinia triticina : ~10 days at 20degC
`nipd`	: infectious period (in degree.day) - default value for Puccinia triticina : ~20 days at 20degC
`dmfr`	: daily multiplication factor - default value number of effective spores produced by lesion
`SITE0`	:
`weather`	: weather data.frame for one single year
`sdate`	: starting date
`ldate`	: ending date
`XLAT0`	:

Details

This model is a classical SEIR model proposed by Zadoks (1971) to simulate epidemics of diseases of crops. It is a Susceptible-Exposed-Infectious-Removed (SEIR) model. This simple model of an epidemic is based on the epidemiological concepts "latent period", "infectious period", and "multiplication factor". The crop is considered to consist of a large but finite number of infectious sites. The physical dimensions of an infectious site roughly coincide with the reproductive unit of the parasite studied. Different pathosystems (with different infectious site definitions) can be considered with this model. A full description, is available in the original paper: The model has four essential state variables representing the number of sites in each state XVAC for vacant (healthy) sites, XLAT for latent site, XINF for infectant sites and XCTR for the cumulative total of removal (post infectious) sites. Two supplementary variables based on the state variables are used defined as XTO1 = XLAT+XINF+XCTR and XSEV = XINF+XCTR. Fluxes or rates between the state variables are defined as rocc for occupation, rapp for apparition and rrem for removal. The model has a time step of one day (dt=1). The system modeled is one hectare of a wheat crop.

Value

list with a data.frame with daily day, DACE, XVAC, XLAT, XINF, XCTR ,XTO1, XSEV= XSEV, severity and a vector of parameter value (nlpd, nipd, dmfr, SITE0).

Source

Script written from equation described in Zadoks, J.C. 1971. Systems Analysis and the Dynamics of Epidemics. Phytopathology. 61:441-598.

See Also

epirice.model

Examples

```
weather=subset(weather_FranceWest, WEYR==1997 & idsite==39)
out=zakoks.original.model(nlpd=4*10,nipd=1*10,dmfr=16,SITE0 = 5*10^9,
weather, sdate = 145, ldate = 145+50 , XLAT0=1)
plot(out$sim$DACE,out$sim$severity, type="l")
```

Index