adult-interface.R

doree

2024-07-24

# generic methods for adult component

@title Derivatives for adult mosquitoes @description This method dispatches on the type of pars$MYZpar. @param t current simulation time @param y state vector @param pars an xds object @param s the species index @return the derivatives a [vector] @export

dMYZdt <- function(t, y, pars, s) {  
 UseMethod("dMYZdt", pars$MYZpar[[s]])  
}

@title Compute the steady states as a function of the daily EIR @description This method dispatches on the type of MYZpar. @param Lambda the daily emergence rate of adult mosquitoes @param kappa net infectiousness @param MYZpar a list that defines an adult model @return none @export

xde\_steady\_state\_MYZ = function(Lambda, kappa, MYZpar){  
 UseMethod("xde\_steady\_state\_MYZ", MYZpar)  
}

@title Compute the steady states as a function of the daily EIR @description This method dispatches on the type of MYZpar. @param Lambda the daily emergence rate of adult mosquitoes @param MYZpar a list that defines an adult model @return none @export

xde\_steady\_state\_M = function(Lambda, MYZpar){  
 UseMethod("xde\_steady\_state\_M", MYZpar)  
}

@title A function to set up adult mosquito models @description This method dispatches on MYZname. @param MYZname the name of the model @param pars a [list] @param s the species index @param EIPopts is a [list] @param MYZopts a [list] @param calK is a [matrix] @return [list] @export

xde\_setup\_MYZpar = function(MYZname, pars, s, EIPopts, MYZopts=list(), calK=diag(1)){  
 class(MYZname) <- MYZname  
 UseMethod("xde\_setup\_MYZpar", MYZname)  
}

@title Derivatives for adult mosquitoes @description This method dispatches on the type of pars$MYZpar. @param t current simulation time @param y state vector @param pars a [list] @param s the species index @return the derivatives a [vector] @export

DT\_MYZt <- function(t, y, pars, s) {  
 UseMethod("DT\_MYZt", pars$MYZpar[[s]])  
}

@title Compute the steady states as a function of the daily EIR @description This method dispatches on the type of MYZpar. @inheritParams xde\_steady\_state\_MYZ @return none @export

dts\_steady\_state\_MYZ = function(Lambda, kappa, MYZpar){  
 UseMethod("dts\_steady\_state\_MYZ", MYZpar)  
}

@title A function to set up adult mosquito models @description This method dispatches on MYZname. @param MYZname the name of the model @param pars a [list] @param s the species index @param EIPopts is a [list] @param MYZopts a [list] @param calK is a [matrix] @return [list] @export

dts\_setup\_MYZpar = function(MYZname, pars, s, EIPopts, MYZopts=list(), calK=diag(1)){  
 class(MYZname) <- MYZname  
 UseMethod("dts\_setup\_MYZpar", MYZname)  
}

@title Set bloodfeeding and mortality rates to baseline @description This method dispatches on the type of pars$MYZpar. It should set the values of the bionomic parameters to baseline values. @param t current simulation time @param y state vector @param pars a [list] @param s the species index @return a [list] @export

MBionomics <- function(t, y, pars, s) {  
 UseMethod("MBionomics", pars$MYZpar[[s]])  
}

@title Time spent host seeking/feeding and resting/ovipositing @description This method dispatches on the type of pars$MYZpar. @param t current simulation time @param pars a [list] @return either a [numeric] vector if the model supports this feature, or [NULL] @export

F\_tau <- function(t, pars) {  
 UseMethod("F\_tau", pars$MYZpar)  
}

@title Blood feeding rate of the infective mosquito population @description This method dispatches on the type of pars$MYZpar. @param t current simulation time @param y state vector @param pars a [list] @param s the species index @return a [numeric] vector of length nPatches @export

F\_fqZ <- function(t, y, pars, s) {  
 UseMethod("F\_fqZ", pars$MYZpar[[s]])  
}

@title Blood feeding rate of the mosquito population @description This method dispatches on the type of pars$MYZpar. @param t current simulation time @param y state vector @param pars a [list] @param s the species index @return a [numeric] vector of length nPatches @export

F\_fqM <- function(t, y, pars, s) {  
 UseMethod("F\_fqM", pars$MYZpar[[s]])  
}

@title Number of eggs laid by adult mosquitoes @description This method dispatches on the type of pars$MYZpar. @param t current simulation time @param y state vector @param pars a [list] @param s the species index @return a [numeric] vector of length nPatches @export

F\_eggs <- function(t, y, pars, s) {  
 UseMethod("F\_eggs", pars$MYZpar[[s]])  
}

@title Return the variables as a list @description This method dispatches on the type of pars$MYZpar[[s]]. @param y the variables @param pars a [list] @param s the vector species index @return a [list] @export

list\_MYZvars <- function(y, pars, s) {  
 UseMethod("list\_MYZvars", pars$MYZpar[[s]])  
}

@title Put MYZvars in place of the MYZ variables in y @description This method dispatches on the type of pars$MYZpar[[s]]. @param MYZvars the variables @param y the variables @param pars a [list] @param s the vector species index @return a [list] @export

put\_MYZvars <- function(MYZvars, y, pars, s) {  
 UseMethod("put\_MYZvars", pars$MYZpar[[s]])  
}

@title A function to set up adult mosquito models @description This method dispatches on MYZname. @param pars a [list] @param s the species index @param MYZopts a [list] @return [list] @export

setup\_MYZinits = function(pars, s, MYZopts=list()){  
 UseMethod("setup\_MYZinits", pars$MYZpar[[s]])  
}

@title Add indices for adult mosquitoes to parameter list @description This method dispatches on the type of pars$MYZpar. @param pars a [list] @param s the species index @return [list] @export

make\_indices\_MYZ <- function(pars, s) {  
 UseMethod("make\_indices\_MYZ", pars$MYZpar[[s]])  
}

@title Parse the outputs and return the variables by name in a list @description This method dispatches on the type of pars$MYZpar. It computes the variables by name and returns a named list. @param outputs a [matrix] of outputs from deSolve @param pars a [list] that defines a model @param s the species index @return [list] @export

parse\_outputs\_MYZ <- function(outputs, pars, s) {  
 UseMethod("parse\_outputs\_MYZ", pars$MYZpar[[s]])  
}

@title Return initial values as a vector @description This method dispatches on the type of pars$MYZpar. @param pars a [list] @param s the species index @return [numeric] @export

get\_inits\_MYZ <- function(pars, s) {  
 UseMethod("get\_inits\_MYZ", pars$MYZpar[[s]])  
}

@title Set the initial values as a vector @description This method dispatches on the type of pars$MYZpar. @param pars a [list] @param y0 a vector of variable values from a simulation @param s the species index @return a [list] @export

update\_inits\_MYZ <- function(pars, y0, s) {  
 UseMethod("update\_inits\_MYZ", pars$MYZpar[[s]])  
}

@title Convert a model from dde to the corresponding ode @description This method dispatches on the type of pars$MYZpar$xde @param pars a [list] @return a [list] @export

dde2ode\_MYZ = function(pars){  
 UseMethod("dde2ode\_MYZ", pars$MYZpar$xde)  
}

@title Convert a model from dde to the corresponding ode @description If it is already an ode, return pars unchanged. @param pars a [list] @return a [list] @export

dde2ode\_MYZ.ode = function(pars){pars}

@title Convert a model from dde to the corresponding ode @description If it is a dde, return the corresponding ode @param pars a [list] @return a [list] @export

dde2ode\_MYZ.dde = function(pars){  
 pars$MYZpar$xde <- "ode"  
 pars$MYZpar$solve\_as <- "ode"  
 pars <- xde\_make\_MYZpar\_RM(pars, MYZopts<- pars$MYZpar,  
 calK=pars$MYZpar$calK)  
 pars <- make\_indices(pars)  
 return(pars)  
}