

NicheMapR and DEB theory II: the DEB models

Michael Kearney

2023-06-09

Contents

Overview	1
Read in allstat.mat to R	1
Loading the allstat file into R and extracting data	2
Run a DEB simulation for a particular species	5
Run a DEB simulation with the NicheMapR ectotherm model	7
References	12

Overview

This document demonstrates the Dynamic Energy Budget model functions of the NicheMapR package. It starts by showing how to read in the allstat.mat file into R. It then demonstrates an R function from the NicheMapR package that integrates the DEB model through time given temperature and food, computing a variety of different outputs, and can be run for a given species in the allstat file.

Finally it runs the ectotherm model of NicheMapR with the DEB module turned on to compute the ‘thermodynamic niche’, i.e. the integrated heat, water and nutritional budget and the consequences for development, growth and reproduction across the whole life cycle in a realistic environment.

Note that the latter simulations assume part I has been run and a microclimate scenario was saved from that run.

Read in allstat.mat to R

The AmPtool github repository houses the file ‘allStat.mat’, which includes parameters and implied properties for all species in the AmP collection (Marques et al. 2018). This file can be read into R with the ‘R.matlab’ package and saved as an R data object. The initial read of the .mat file takes a while (5-10 minutes), but the ‘Rda’ file to which it is converted loads much faster (seconds).

Here is the code to read and convert the ‘allStat.mat’ file.

```
# install.packages('R.matlab')
library(R.matlab)
allStat <- readMat("allStat.mat") # this will take a few mintues
save(allStat, file = "allstat.Rda") # save it as an R data file for faster future loading
```

Loading the allstat file into R and extracting data

Now this file can be read into memory in R and manipulated to extract parameter values for different species or taxa.

The file is a list of lists. For example, you can get a list and count of all the species in the collection.

```
library(knitr) # this packages has a function for producing formatted tables.
load("allStat.Rda")

allDEB.species <- unlist(labels(allStat$allStat)) # get all the species names
allDEB.species <- allDEB.species[1:(length(allDEB.species) -
  2)] # last two elements are not species names
kable(head(allDEB.species))
```

x
Haliclona.oclata
Metridium.senile
Ptilosarcus.gurneyi
Chironex.fleckeri
Hydra.viridissima
Pelagia.noctiluca

```
Nspecies <- length(allStat$allStat)
Nspecies
```

```
## [1] 4036
```

And you can extract all parameters for a given species. Here it is done by using the ‘assign’ function. First, the slot in the top-level list of species is found which matches the species name chosen. Then, the parameter names for that entry are extracted. Finally, all elements of the lists of data for that species are extracted and assigned names corresponding to the parameter names just extracted, using the ‘assign’ function, using a for loop. Once it has run, all the parameters, implied properties and details about the entry are thus loaded into the memory with appropriate names, ready for use.

```
species <- "Anolis.carolinensis"
species.slot <- which(allDEB.species == species)
par.names <- unlist(labels(allStat$allStat[[species.slot]]))

for (i in 1:length(par.names)) {
  assign(par.names[i], unlist(allStat$allStat[[species.slot]][i]))
}
```

Here’s the full list of par names extracted for *Anolis carolinensis*.

```
par.names
```

```
## [1] "species"      "species.en"  "family"      "order"       "class"
## [6] "phylum"     "id.CoL"     "ecoCode"     "model"       "MRE"
## [11] "SMSE"        "COMPLETE"   "data"        "author"      "date.subm"
## [16] "date.acc"    "T.ref"      "T.typical"   "T.A"         "z"
```

```
## [21] "F.m"      "kap.X"    "kap.P"    "v"        "kap"
## [26] "kap.R"    "p.M"      "p.T"      "k.J"      "E.G"
## [31] "E.Hb"     "E.Hp"     "h.a"      "s.G"      "d.X"
## [36] "d.V"      "d.E"      "d.P"      "mu.X"     "mu.V"
## [41] "mu.E"     "mu.P"     "mu.C"     "mu.H"     "mu.O"
## [46] "mu.N"     "n.CX"     "n.HX"     "n.OX"     "n.NX"
## [51] "n.CV"     "n.HV"     "n.OV"     "n.NV"     "n.CE"
## [56] "n.HE"     "n.OE"     "n.NE"     "n.CP"     "n.HP"
## [61] "n.OP"     "n.NP"     "n.CC"     "n.HC"     "n.OC"
## [66] "n.NC"     "n.CH"     "n.HH"     "n.OH"     "n.NH"
## [71] "n.CO"     "n.HO"     "n.OO"     "n.NO"     "n.CN"
## [76] "n.HN"     "n.ON"     "n.NN"     "f"        "T"
## [81] "c.T"      "p.Am"     "w.X"      "w.V"      "w.E"
## [86] "w.P"      "M.V"      "y.V.E"    "y.E.V"    "k.M"
## [91] "k"        "E.m"      "m.Em"     "g"        "L.m"
## [96] "L.T"      "l.T"      "ome"      "J.E.Am"   "y.E.X"
## [101] "y.X.E"    "p.Xm"     "J.X.Am"   "y.P.X"    "y.X.P"
## [106] "y.P.E"    "eta.XA"   "eta.PA"   "eta.VG"   "J.E.M"
## [111] "J.E.T"    "j.E.M"    "j.E.J"    "kap.G"    "E.V"
## [116] "K"        "M.Hb"     "U.Hb"     "V.Hb"     "u.Hb"
## [121] "v.Hb"     "M.Hp"     "U.Hp"     "V.Hp"     "u.Hp"
## [126] "v.Hp"     "t.E"      "t.starve" "s.M"      "r.B"
## [131] "W.dWm"    "dWm"      "U.E0"     "E.0"      "M.E0"
## [136] "Wd.0"     "Ww.0"     "l.b"      "L.b"      "M.Vb"
## [141] "del.Ub"   "Ww.b"     "Wd.b"     "E.Wb"     "a.b"
## [146] "g.Hb"     "s.Hbp"    "s.HLbp"   "l.p"      "L.p"
## [151] "M.Vp"     "M.Ep"     "Ww.p"     "Wd.p"     "E.Wp"
## [156] "a.p"      "g.Hp"     "s.s"      "l.i"      "L.i"
## [161] "M.Vi"     "M.Ei"     "Ww.i"     "Wd.i"     "E.Wi"
## [166] "xi.WE"    "del.Wb"   "del.Wp"   "del.V"    "h.W"
## [171] "h.G"      "S.b"      "S.p"      "a.m"      "a.99"
## [176] "R.i"      "N.i"      "M.E0.min.G" "M.E0.min.R" "eb.min.G"
## [181] "eb.min.R" "ep.min"   "sM.min"   "p.Xb"     "J.Xb"
## [186] "F.mb"     "p.Xp"     "J.Xp"     "F.mp"     "p.Xi"
## [191] "J.Xi"     "F.mi"     "p.Ab"     "p.Cb"     "p.Sb"
## [196] "p.Jb"     "p.Gb"     "p.Rb"     "p.Db"     "p.Ap"
## [201] "p.Cp"     "p.Sp"     "p.Jp"     "p.Gp"     "p.Rp"
## [206] "p.Dp"     "p.Ai"     "p.Ci"     "p.Si"     "p.Ji"
## [211] "p.Gi"     "p.Ri"     "p.Di"     "J.Cb"     "J.Cp"
## [216] "J.Ci"     "J.Hb"     "J.Hp"     "J.Hi"     "J.Ob"
## [221] "J.Op"     "J.Oi"     "J.Nb"     "J.Np"     "J.Ni"
## [226] "RQ.b"     "UQ.b"     "WQ.b"     "SDA.b"    "VO.b"
## [231] "p.Tb"     "RQ.p"     "UQ.p"     "WQ.p"     "SDA.p"
## [236] "VO.p"     "p.Tp"     "RQ.i"     "UQ.i"     "WQ.i"
## [241] "SDA.i"    "VO.i"     "p.Ti"     "1"        "1"
```

And here's an example of plotting the implied mass-specific oxygen consumption rate as a function of dry mass for all the squamata (lizards and snakes) in the collection.

```
pars.wanted <- c("Wd.i", "VO.i") # choose parameters of interest

# make an empty matrix to put the values in
all.VOi <- matrix(NA, nrow = length(allDEB.species), ncol = length(pars.wanted))
```

```

# loop through all species, get the parameters
for (i in 1:length(allDEB.species)) {
  par.names <- unlist(labels(allStat$allStat[[i]]))
  for (j in 1:length(pars.wanted)) {
    par.slot <- which(par.names == pars.wanted[j])
    all.VOi[i, j] <- unlist(allStat$allStat[[i]][par.slot])
  }
}

# get just the values for a given order
order.slot <- which(par.names == "order") # find which item contains the taxonomic order
order.wanted <- "Squamata" # choose order of interest

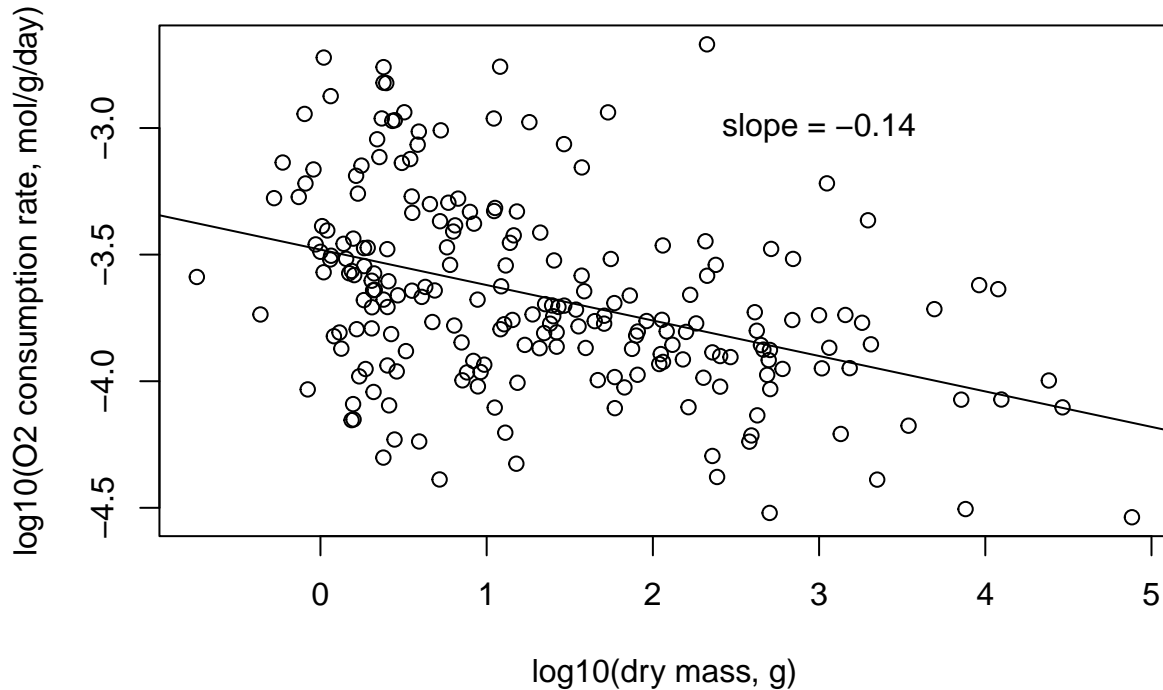
# loop through all species and find those matching the
# chosen order
all.orders <- vector()
for (i in 1:length(allDEB.species)) {
  all.orders[i] <- unlist(allStat$allStat[[c(i, order.slot)]])
}

# subset the extracted parameters for just those species in
# the chosen order
squam.voi <- all.VOi[which(all.orders %in% order.wanted), ]

# plot mass-specific metabolic rate vs mass (log10
# transformed)
plot(log10(squam.voi[, 1]), log10(squam.voi[, 2] * -1), ylab = "log10(O2 consumption rate, mol/g/day)",
      xlab = "log10(dry mass, g)")

# estimate the slope and plot it
estW02 <- coef(summary(lm(log10(squam.voi[, 2] * -1) ~ log10(squam.voi[,
  1]))))
slp <- estW02[2]
abline(estW02[1], estW02[2])
text(3, -3, paste0("slope = ", round(slp, 2)))

```



Run a DEB simulation for a particular species

The NicheMapR package has DEB functions that integrate the DEB model through time as a function of body temperature and food. There are three DEB functions - 'DEB', 'DEB_const', 'DEB_var' and 'DEB_euler'. The latter uses a simple integration where the rates of change of the state variables (i.e. structure, reserve, maturity, reproduction buffer) are assumed to be constant within a time period and the state at a given time is simply the state at time period before plus the rate of change for the current time period. This integration method is fast but inaccurate for large step sizes/rapid environmental changes. The other DEB function uses the deSolve package to integrate through time, specifically the ode45 method, which dynamically adjusts the step size according to how fast the system is evolving, and is far more accurate but more computationally intensive. The functions include a stomach model, which requires a parameter for the maximum structure-specific stomach energy density. It also includes a reproduction batch model as originally described in Pequerie et al. (2009), which requires input on the clutch size.

A wrapper function called 'rundeb' calls either of these DEB routines for a specified species for a specified duration and sequence of body temperatures and food levels. The step size can be given via parameter 'div', which determines how frequently output is provided (and dictates the accuracy of the DEB_euler function if used).

The 'rundeb' function makes some plots by default of growth, food and respiration. It also allows some exploration of the effects of varying parameters, specifically the reproduction allocation parameter kappa, somatic maintenance costs, initial egg size/energy content and overall size. The latter, controlled by the factor 'z.mult', causes scaling to occur according to the DEB theory covariation rules, whereby size at birth, size at puberty, ultimate size, egg size and longevity all covary in a particular manner.

```

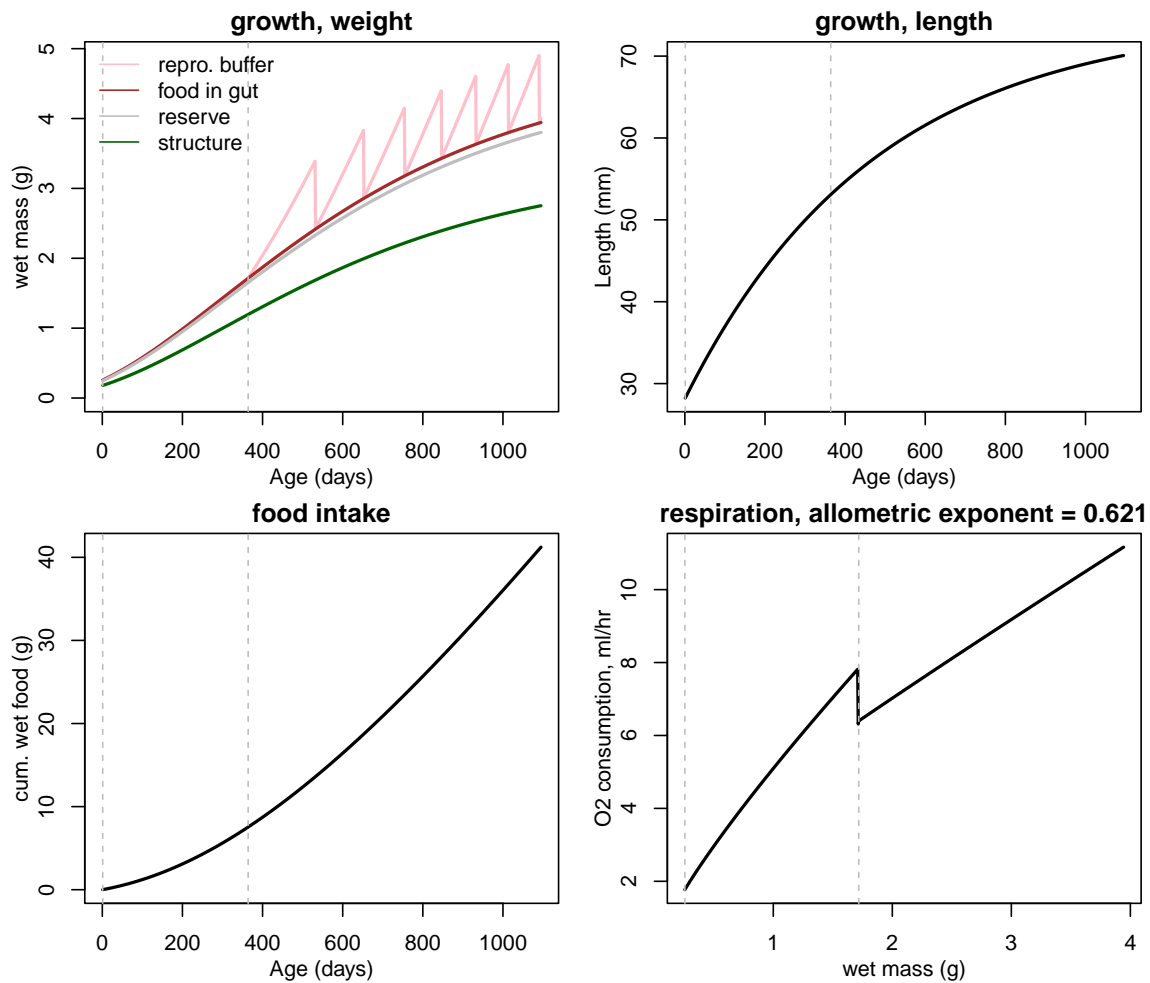
library(NicheMapR)
species <- "Anolis.carolinensis" # must be in the AmP collection - see allDEB.species list
ndays <- 365 * 3 # number days to run the simulation for
div <- 1 # time step divider (1 = days, 24 = hours, etc.) - keep small if using Euler method
Tbs <- rep(27, ndays * div) # °C, body temperature
starvetime <- 0 # length of low food period when simulating starvation
X <- 100 # J/cm2 base food density
Xs <- c(rep(X, ndays * div/2), rep(5e-06, starvetime), rep(X,
  ndays * div/2 - starvetime + 1)) # food density (J/cm2 or J/cm3)
E_sm <- 350 # J/cm3, volume-specific stomach energy
clutchsize <- 2 # -, clutch size
kap.mult <- 1 # -, factor to multiply kappa by (caps at 1)
p.M.mult <- 1 # -, factor to multiply p.M by (caps at 1)
v.mult <- 1 # -, factor to multiply energy conductance by
z.mult <- 1 # -, factor to multiply zoom factor by
E.O.mult <- 1 # -, factor by which to multiply initial egg energy

mass.unit <- "g"
length.unit <- "mm"
Euler <- 0 # use Euler integration (faster but less accurate) (1) or deSolve's ODE solve (0)
plot <- 1 # plot results?
start.stage <- 1 # stage in life cycle to start (0 = egg, 1 = juvenile, 2 = puberty)

deb <- rundeb(species = species, ndays = ndays, div = div, Tbs = Tbs,
  clutchsize = clutchsize, kap.mult = kap.mult, v.mult = v.mult,
  p.M.mult = p.M.mult, Xs = Xs, z.mult = z.mult, E.O.mult = E.O.mult,
  mass.unit = mass.unit, length.unit = length.unit, start.stage = start.stage,
  E_sm = E_sm, Euler = Euler, plot = plot)

```

NOTE: No length data in parameter estimation, assuming del.M of 0.2, predictions of physical lengths



Run a DEB simulation with the NicheMapR ectotherm model

Finally, we can put the microclimate, heat budget and DEB model together by turning on the DEB model in the NicheMapR ectotherm function (Kearney and Porter, 2020) and passing it the DEB parameters for the species of interest - here we use *Anolis carolinensis* but you can put any species you like in there and see how it would grow at the site the microclimate was simulated for (in this case Baton Rouge - as simulated in part I of this workshop). Note that del.M was not estimated for this species so the value for *Anolis gundlachi* was used.

```
load("micro_baton_rouge_2017_2020.Rda") # load the microclimate simulation from part I
load("allstat.Rda") # load the allstat file

species <- "Anolis.carolinensis"

allDEB.species <- unlist(labels(allStat$allStat)) # get all the species names
allDEB.species <- allDEB.species[1:(length(allDEB.species) -
  2)] # last two elements are not species
species.slot <- which(allDEB.species == species)
```

```

par.names <- unlist(labels(allStat$allStat[[species.slot]]))
# clear possible missing parameters
if (exists("E.Hj") == TRUE) {
  rm(E.Hj)
}
if (exists("E.He") == TRUE) {
  rm(E.He)
}
if (exists("L.j") == TRUE) {
  rm(L.j)
}
if (exists("T.L") == TRUE) {
  rm(T.L)
}
if (exists("T.H") == TRUE) {
  rm(T.H)
}
if (exists("T.AL") == TRUE) {
  rm(T.AL)
}
if (exists("T.AH") == TRUE) {
  rm(T.AH)
}

for (i in 1:length(par.names)) {
  assign(par.names[i], unlist(allStat$allStat[[species.slot]][i]))
}
# assign possible missing parameters
if (exists("E.Hj") == FALSE) {
  E.Hj <- E.Hb
}
if (exists("E.He") == FALSE) {
  E.He <- E.Hb
}
if (exists("L.j") == FALSE) {
  L.j <- L.b
}

p.Xm <- p.Am/kap.X # redefining F.m to max possible value
del.M <- 0.2131 # from Anolis gundlachi

# morph, behav and water loss
shape <- 3 # animal shape - 3 = lizard
alpha_max <- 0.85 # maximum solar absorptivity
alpha_min <- 0.8 # minimum solar absorptivity
T_RB_min <- 15 # min Tb at which they will attempt to leave retreat
T_B_min <- 20 # min Tb at which leaves retreat to bask
T_F_min <- 25 # minimum Tb at which activity occurs
T_F_max <- 35 # maximum Tb at which activity occurs
T_pref <- 33.8 # preferred Tb (will try and regulate to this)
CT_max <- 42 # critical thermal minimum (affects choice of retreat)
CT_min <- 9 # critical thermal maximum (affects choice of retreat)
mindepth <- 2 # min depth (node, 1-10) allowed
maxdepth <- 10 # max depth (node, 1-10) allowed

```



```

shade_seek <- 1 # shade seeking?
burrow <- 1 # can it burrow?
climb <- 0 # can it climb to thermoregulate?
nocturn <- 0 # nocturnal activity
crepus <- 0 # crepuscular activity
diurn <- 1 # diurnal activity
pct_wet <- 0.1 # percent of surface area acting like a free water surface
z.mult <- 1 # body size scaling parameter

# DEB initial state and misc.

# egg V_init <- 3e-9 E_init <- E.0 / V_init E_H_init <- 0
# stage <- 0

# hatchling
V_init <- L.b^3
E_init <- E.m
E_H_init <- E.Hb
stage <- 1

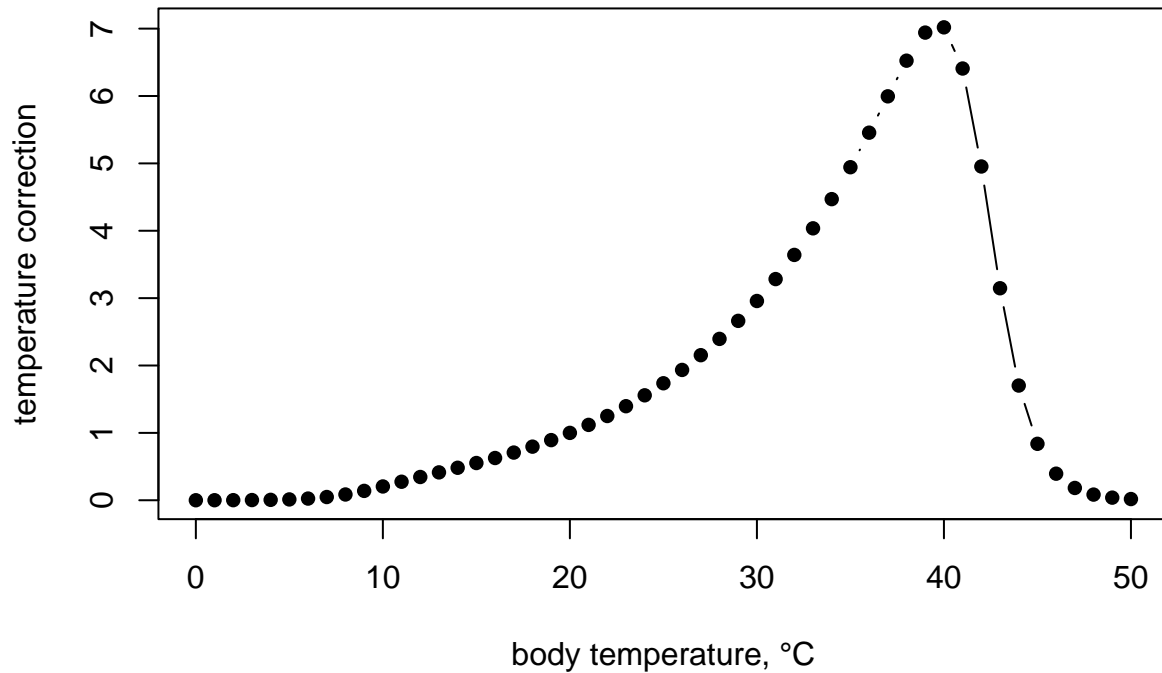
# mature V_init <- L.p^3 E_init <- E.m E_H_init <- E.Hp+2
# stage <- 2

# assign missing 5-par thermal response curve parameters if
# necessary
if (exists("T.L") == FALSE) {
  T.L <- CT_min + 273.15
}
if (exists("T.H") == FALSE) {
  T.H <- CT_max + 273.15
}
if (exists("T.AL") == FALSE) {
  T.AL <- 50000
}
if (exists("T.AH") == FALSE) {
  T.AH <- 90000
}

# plot thermal response curve, out of interest

get_c_T <- function(T_A, T_L, T_H, T_AL, T_AH, T_REF, T_b) {
  # temperature correction function
  c_T <- exp(T_A/(T_REF - T_A/T_b)) * (1 + exp(T_AL/T_REF -
    T_AL/T_L) + exp(T_AH/T_H - T_AH/T_REF))/(1 + exp(T_AL/T_b -
    T_AL/T_L) + exp(T_AH/T_H - T_AH/T_b))
}
T_b <- seq(0, 50) + 273.15 # sequence of body temperatures, K
c_Ts <- get_c_T(T_A = T.A, T_L = T.L, T_H = T.H, T_AL = T.AL,
  T_AH = T.AH, T_REF = T.ref, T_b = T_b) # compute temperature correction
plot(T_b - 273.15, c_Ts, type = "b", ylab = "temperature correction",
  xlab = "body temperature, °C", pch = 16)

```



```
# reproduction parameters
viviparous <- 0 # live bearing (1) or egg laying (0)
clutchsize <- 2 # how many eggs per clutch?
photostart <- 3 # winter solstice is the start of the reproduction cycle
photofinish <- 2 # autumnal equinox is the end of the reproduction cycle

# run the ectotherm model

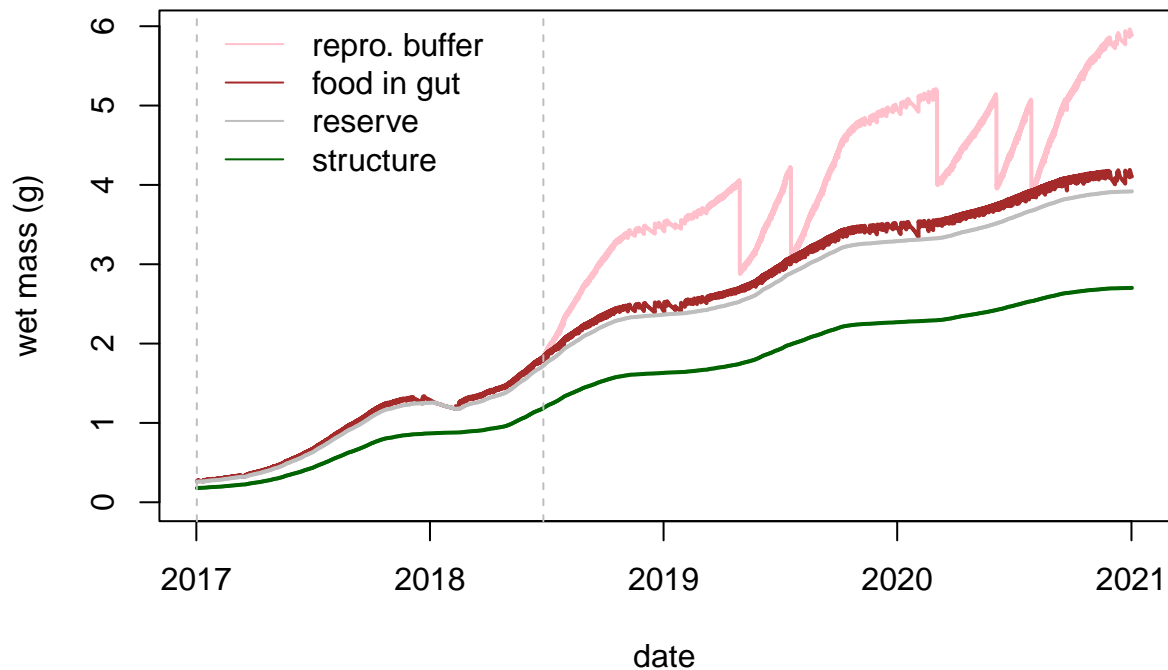
ecto <- ectotherm(DEB = 1, viviparous = viviparous, clutchsize = clutchsize,
  z.mult = z.mult, shape = shape, alpha_max = alpha_max, alpha_min = alpha_min,
  T_F_min = T_F_min, T_F_max = T_F_max, T_B_min = T_B_min,
  T_RB_min = T_RB_min, T_pref = T_pref, CT_max = CT_max, CT_min = CT_min,
  diurn = diurn, nocturn = nocturn, crepus = crepus, shade_seek = shade_seek,
  burrow = burrow, climb = climb, mindepth = mindepth, maxdepth = maxdepth,
  pct_wet = pct_wet, z = z * z.mult, del_M = del.M, p_Xm = p.Xm,
  kap_X = kap.X, v = v/24, kap = kap, p_M = p.M/24, E_G = E.G,
  kap_R = kap.R, k_J = k.J/24, E_Hb = E.Hb * z.mult^3, E_Hj = E.Hj *
    z.mult^3, E_Hp = E.Hp * z.mult^3, E_He = E.He * z.mult^3,
  h_a = h.a/(24^2), s_G = s.G, T_REF = T.ref, T_A = T.A, T_AL = T.AL,
  T_AH = T.AH, T_L = T.L, T_H = T.H, E_0 = E.0 * z.mult^4,
  f = f, d_V = d.V, d_E = d.E, d_Egg = d.E, mu_X = mu.X, mu_E = mu.E,
  mu_V = mu.V, mu_P = mu.P, kap_X_P = kap.P, n_X = c(n.CX,
    n.HX, n.OX, n.ON), n_E = c(n.CE, n.HE, n.OE, n.OE), n_V = c(n.CV,
    n.HV, n.OV, n.OV), n_P = c(n.CP, n.HP, n.OP, n.OP), n_M_nitro = c(n.NC,
    n.NH, n.NO, n.NN), L_b = L.b, V_init = V_init, E_init = E_init,
  E_H_init = E_H_init, stage = stage, photostart = photostart,
```

```

    photofinish = photofinish)

# retrieve output
environ <- as.data.frame(ecto$environ) # behaviour, Tb and environment
enbal <- as.data.frame(ecto$enbal) # heat balance outputs
masbal <- as.data.frame(ecto$masbal) # mass balance outputs
debout <- as.data.frame(ecto$debout) # DEB model outputs
par(mfrow = c(1, 1))
plot(micro$dates, debout$WETMASS, type = "l", xlab = "date",
     ylab = paste0("wet mass (g)"), col = "pink", lwd = 2, ylim = c(0,
     max(debout$WETMASS)))
points(micro$dates, debout$V, type = "l", xlab = "date", ylab = paste0("wet mass (g)"),
      col = "dark green", lwd = 2)
points(micro$dates, debout$WETMASS - debout$WETGONAD, type = "l",
      lwd = 2, col = "brown")
points(micro$dates, debout$WETMASS - debout$WETGONAD - debout$WETGUT,
      type = "l", lwd = 2, col = "grey")
abline(v = micro$dates[which(debout$E_H > E.Hb)[1]], lty = 2,
      col = "grey")
abline(v = micro$dates[which(debout$E_H > E.Hp)[1]], lty = 2,
      col = "grey")
legend(micro$dates[1], max(debout$WETMASS) * 1.05, c("repro. buffer",
      "food in gut", "reserve", "structure"), lty = rep(1, 4),
      col = c("pink", "brown", "grey", "dark green"), bty = "n")

```



References

- Kearney, M. R., & Porter, W. P. (2020). NicheMapR – an R package for biophysical modelling: The ectotherm and Dynamic Energy Budget models. *Ecography*, 43(1), 85–96. <https://doi.org/10.1111/ecog.04680>
- Marques G. M., Augustine S., Lika K., Pecquerie L., Domingos T. & Kooijman S. A. L. M. (2018) The AmP project: Comparing species on the basis of dynamic energy budget parameters. *PLOS Computational Biology* 14 , e1006100.
- Pecquerie L., Petitgas P. & Kooijman S. A. L. M. (2009) Modeling fish growth and reproduction in the context of the Dynamic Energy Budget theory to predict environmental impact on anchovy spawning duration. *Journal of Sea Research* 62 , 93–105.