Appendices for ‘An individual-based model of ectotherm movement integrating metabolic and microclimatic constraints’

Matthew Malishev1,2\*, C. Michael Bull3, & Michael R. Kearney2

*1 Centre of Excellence for Biosecurity Risk Analysis, 2 School of BioSciences, University of Melbourne, Parkville, Melbourne, 3010, Australia* *3 School of Biological Sciences, Flinders University, Adelaide, 5001, Australia*[[1]](#footnote-1)

\*Corresponding author: [matthew.malishev@gmail.com](mailto:matthew.malishev@gmail.com)

**Table of Contents**

Data collection 4

NicheMapR microclimate model overview 5

onelump\_varenv.R. 6

DEB.R. 12

Appendix 1 20

space and time scales 20

interface 20

globals 20

turtles-own 21

patches-own 22

breeds 22

setup 22

go 24

update *Tb* 25

make-decision 25

search 28

bounce 29

handle food 29

shade search 29

rest 30

socialise 30

update food levels 30

report patch color 30

report patch type 31

report results 31

to save world 32

spatial plot 32

get home range 32

draw home range 32

Appendix 2 34

Initial setup 34

Read in microclimate data 35

Read in DEB parameters 36

Initialise decision-making and DEB models 37

Run simulation 39

Example of data output files from simulation 44

Appendix 3 45

Summary of DEB parameters and primary metabolic pathways. 45

Figures 48

References 50

##### 

# Data collection

All data were collected at the sleepy lizard habitat study site (139°21’E, 33°55’S) at the Bundey Bore field station in the mid-north of South Australia during the breeding season (September to December, 2009). Animal data are for the adult sleepy lizard (n = 60). Individual animals were tagged with GPS units, step counters (‘waddleometers’), and skin surface temperature probes at the beginning of the breeding season and tracked throughout the season using radio telemetry. Animals were captured and GPS data downloaded every two weeks throughout the breeding season for each individual, with batteries for the units replaced when needed. GPS units reported locations every 10 minutes, waddleometers recorded step counts every 2 minutes, and temperature probes recorded skin surface temperature every 2 minutes.

The simulation model uses a 2-minute time step to correspond to the frequency of observed data.

##### 

# NicheMapR microclimate model overview

The NicheMapR microclimate model calculates hourly estimates of solar and infrared radiation, air temperature at 1 m and 1 cm above ground level, wind velocity, relative humidity, and soil temperature at different intervals, e.g. 0 cm, 10 cm, 20 cm, 50 cm, 100 cm, and 200 cm. The model uses minimum and maximum daily air temperature, wind speed, relative humidity, soil properties (conductivity, specific heat, density, solar reflectivity, emissivity), as well as the roughness height, slope, and aspect. Climatic data are gathered from a global data set of monthly mean daily minimum and maximum air temperatures and monthly mean daily humidity and wind speeds. Soil surface temperatures are computed using heat balance equations, accounting for heat exchange via radiation, convection, conduction, and evaporation.

For simulation time steps, the microclimate model verifies the microclimate conditions for the current simulation hour of the day, e.g. noon or 18:00, and location in space, i.e. the study site for the observed animal data, and updates patches in the simulation landscape (either sun or shade) with these microenvironment conditions. As the simulated animal moves in or out of these patches at each time step, the animal updates its current , including rates of change in per 2-minute time step.

The onelump\_varenv.R and DEB.R functions update the individual internal thermal and metabolic states, respectively. See below for both model functions.

##### 

### onelump\_varenv.R.

onelump\_varenv.R available on [**Github**](https://github.com/darwinanddavis/MalishevBullKearney/blob/master/onelump_varenv.R).

onelump\_varenv<-function (t = seq(1, 3600, 60), time = 0, Tc\_init = 5, thresh = 29,   
 AMASS = 500, lometry = 2, Tairf = Tairfun, Tradf = Tradfun,   
 velf = velfun, Qsolf = Qsolfun, Zenf = Zenfun, Flshcond = 0.5,   
 q = 0, Spheat = 3073, EMISAN = 0.95, rho = 932, ABS = 0.85,   
 colchange = 0, lastt = 0, ABSMAX = 0.9, ABSMIN = 0.6, customallom = c(10.4713,   
 0.688, 0.425, 0.85, 3.798, 0.683, 0.694, 0.743), shape\_a = 1,   
 shape\_b = 0.5, shape\_c = 0.5, posture = "n", FATOSK = 0.4,   
 FATOSB = 0.4, sub\_reflect = 0.2, PCTDIF = 0.1, press = 101325)   
{  
 sigma <- 5.67e-08  
 Tair <- Tairf(time + t)  
 vel <- velf(time + t)  
 Qsol <- Qsolf(time + t)  
 Trad <- Tradf(time + t)  
 Zen <- Zenf(time + t)  
 Zenith <- Zen \* pi/180  
 Tc <- Tc\_init  
 Tskin <- Tc + 0.1  
 RHskin <- 100  
 vel[vel < 0.01] <- 0.01  
 abs2 <- ABS  
 if (colchange >= 0) {  
 abs2 <- min(ABS + colchange \* (t - lastt), ABSMAX)  
 }  
 else {  
 abs2 <- max(ABS + colchange \* (t - lastt), ABSMIN)  
 }  
 S2 <- 1e-04  
 DENSTY <- 101325/(287.04 \* (Tair + 273))  
 THCOND <- 0.02425 + (7.038 \* 10^-5 \* Tair)  
 VISDYN <- (1.8325 \* 10^-5 \* ((296.16 + 120)/((Tair + 273) +   
 120))) \* (((Tair + 273)/296.16)^1.5)  
 m <- AMASS/1000  
 C <- m \* Spheat  
 V <- m/rho  
 Qgen <- q \* V  
 L <- V^(1/3)  
 Flshcond <- 0.5  
 if (lometry == 0) {  
 ALENTH <- (V/shape\_b \* shape\_c)^(1/3)  
 AWIDTH <- ALENTH \* shape\_b  
 AHEIT <- ALENTH \* shape\_c  
 ATOT <- ALENTH \* AWIDTH \* 2 + ALENTH \* AHEIT \* 2 + AWIDTH \*   
 AHEIT \* 2  
 ASILN <- ALENTH \* AWIDTH  
 ASILP <- AWIDTH \* AHEIT  
 L <- AHEIT  
 if (AWIDTH <= ALENTH) {  
 L <- AWIDTH  
 }  
 else {  
 L <- ALENTH  
 }  
 R <- ALENTH/2  
 }  
 if (lometry == 1) {  
 R1 <- (V/(pi \* shape\_b \* 2))^(1/3)  
 ALENTH <- 2 \* R1 \* shape\_b  
 ATOT <- 2 \* pi \* R1^2 + 2 \* pi \* R1 \* ALENTH  
 AWIDTH <- 2 \* R1  
 ASILN <- AWIDTH \* ALENTH  
 ASILP <- pi \* R1^2  
 L <- ALENTH  
 R2 <- L/2  
 if (R1 > R2) {  
 R <- R2  
 }  
 else {  
 R <- R1  
 }  
 }  
 if (lometry == 2) {  
 A1 <- ((3/4) \* V/(pi \* shape\_b \* shape\_c))^0.333  
 B1 <- A1 \* shape\_b  
 C1 <- A1 \* shape\_c  
 P1 <- 1.6075  
 ATOT <- (4 \* pi \* (((A1^P1 \* B1^P1 + A1^P1 \* C1^P1 +   
 B1^P1 \* C1^P1))/3)^(1/P1))  
 ASILN <- max(pi \* A1 \* C1, pi \* B1 \* C1)  
 ASILP <- min(pi \* A1 \* C1, pi \* B1 \* C1)  
 S2 <- (A1^2 \* B1^2 \* C1^2)/(A1^2 \* B1^2 + A1^2 \* C1^2 +   
 B1^2 \* C1^2)  
 Flshcond <- 0.5 + 6.14 \* B1 + 0.439  
 }  
 if (lometry == 3) {  
 ATOT <- (10.4713 \* AMASS^0.688)/10000  
 AV <- (0.425 \* AMASS^0.85)/10000  
 ASILN <- (3.798 \* AMASS^0.683)/10000  
 ASILP <- (0.694 \* AMASS^0.743)/10000  
 R <- L  
 }  
 if (lometry == 4) {  
 ATOT = (12.79 \* AMASS^0.606)/10000  
 AV = (0.425 \* AMASS^0.85)/10000  
 ZEN <- 0  
 PCTN <- 1.38171e-06 \* ZEN^4 - 0.000193335 \* ZEN^3 + 0.00475761 \*   
 ZEN^2 - 0.167912 \* ZEN + 45.8228  
 ASILN <- PCTN \* ATOT/100  
 ZEN <- 90  
 PCTP <- 1.38171e-06 \* ZEN^4 - 0.000193335 \* ZEN^3 + 0.00475761 \*   
 ZEN^2 - 0.167912 \* ZEN + 45.8228  
 ASILP <- PCTP \* ATOT/100  
 R <- L  
 }  
 if (lometry == 5) {  
 ATOT = (customallom[1] \* AMASS^customallom[2])/10000  
 AV = (customallom[3] \* AMASS^customallom[4])/10000  
 ASILN = (customallom[5] \* AMASS^customallom[6])/10000  
 ASILP = (customallom[7] \* AMASS^customallom[8])/10000  
 R <- L  
 }  
 if (max(Zen) >= 90) {  
 Qnorm <- 0  
 }  
 else {  
 Qnorm <- (Qsol/cos(Zenith))  
 }  
 if (Qnorm > 1367) {  
 Qnorm <- 1367  
 }  
 if (posture == "p") {  
 Qabs <- (Qnorm \* (1 - PCTDIF) \* ASILP + Qsol \* PCTDIF \*   
 FATOSK \* ATOT + Qsol \* sub\_reflect \* FATOSB \* ATOT) \*   
 abs2  
 }  
 if (posture == "n") {  
 Qabs <- (Qnorm \* (1 - PCTDIF) \* ASILN + Qsol \* PCTDIF \*   
 FATOSK \* ATOT + Qsol \* sub\_reflect \* FATOSB \* ATOT) \*   
 abs2  
 }  
 if (posture == "b") {  
 Qabs <- (Qnorm \* (1 - PCTDIF) \* (ASILN + ASILP)/2 + Qsol \*   
 PCTDIF \* FATOSK \* ATOT + Qsol \* sub\_reflect \* FATOSB \*   
 ATOT) \* abs2  
 }  
 Rrad <- ((Tskin + 273) - (Trad + 273))/(EMISAN \* sigma \*   
 (FATOSK + FATOSB) \* ATOT \* ((Tskin + 273)^4 - (Trad +   
 273)^4))  
 Rrad <- 1/(EMISAN \* sigma \* (FATOSK + FATOSB) \* ATOT \* ((Tc +   
 273)^2 + (Trad + 273)^2) \* ((Tc + 273) + (Trad + 273)))  
 Re <- DENSTY \* vel \* L/VISDYN  
 PR <- 1005.7 \* VISDYN/THCOND  
 if (lometry == 0) {  
 NUfor <- 0.102 \* Re^0.675 \* PR^(1/3)  
 }  
 if (lometry == 3 | lometry == 5) {  
 NUfor <- 0.35 \* Re^0.6  
 }  
 if (lometry == 1) {  
 if (Re < 4) {  
 NUfor = 0.891 \* Re^0.33  
 }  
 else {  
 if (Re < 40) {  
 NUfor = 0.821 \* Re^0.385  
 }  
 else {  
 if (Re < 4000) {  
 NUfor = 0.615 \* Re^0.466  
 }  
 else {  
 if (Re < 40000) {  
 NUfor = 0.174 \* Re^0.618  
 }  
 else {  
 if (Re < 4e+05) {  
 NUfor = 0.0239 \* Re^0.805  
 }  
 else {  
 NUfor = 0.0239 \* Re^0.805  
 }  
 }  
 }  
 }  
 }  
 }  
 if (lometry == 2 | lometry == 4) {  
 NUfor <- 0.35 \* Re^(0.6)  
 }  
 hc\_forced <- NUfor \* THCOND/L  
 GR <- abs(DENSTY^2 \* (1/(Tair + 273.15)) \* 9.80665 \* L^3 \*   
 (Tskin - Tair)/VISDYN^2)  
 Raylei <- GR \* PR  
 if (lometry == 0) {  
 NUfre = 0.55 \* Raylei^0.25  
 }  
 if (lometry == 1 | lometry == 3 | lometry == 5) {  
 if (Raylei < 1e-05) {  
 NUfre = 0.4  
 }  
 else {  
 if (Raylei < 0.1) {  
 NUfre = 0.976 \* Raylei^0.0784  
 }  
 else {  
 if (Raylei < 100) {  
 NUfre = 1.1173 \* Raylei^0.1344  
 }  
 else {  
 if (Raylei < 10000) {  
 NUfre = 0.7455 \* Raylei^0.2167  
 }  
 else {  
 if (Raylei < 1e+09) {  
 NUfre = 0.5168 \* Raylei^0.2501  
 }  
 else {  
 if (Raylei < 1e+12) {  
 NUfre = 0.5168 \* Raylei^0.2501  
 }  
 else {  
 NUfre = 0.5168 \* Raylei^0.2501  
 }  
 }  
 }  
 }  
 }  
 }  
 }  
 if (lometry == 2 | lometry == 4) {  
 Raylei = (GR^0.25) \* (PR^0.333)  
 NUfre = 2 + 0.6 \* Raylei  
 }  
 hc\_free <- NUfre \* THCOND/L  
 hc\_comb <- hc\_free + hc\_forced  
 Rconv <- 1/(hc\_comb \* ATOT)  
 Nu <- hc\_comb \* L/THCOND  
 hr <- 4 \* EMISAN \* sigma \* ((Tc + Trad)/2 + 273)^3  
 hc <- hc\_comb  
 if (lometry == 2) {  
 j <- (Qabs + Qgen + hc \* ATOT \* ((q \* S2)/(2 \* Flshcond) +   
 Tair) + hr \* ATOT \* ((q \* S2)/(2 \* Flshcond) + Trad))/C  
 }  
 else {  
 j <- (Qabs + Qgen + hc \* ATOT \* ((q \* R^2)/(4 \* Flshcond) +   
 Tair) + hr \* ATOT \* ((q \* S2)/(2 \* Flshcond) + Trad))/C  
 }  
 kTc <- ATOT \* (Tc \* hc + Tc \* hr)/C  
 k <- ATOT \* (hc + hr)/C  
 Tcf <- j/k  
 Tci <- Tc  
 Tc <- (Tci - Tcf) \* exp(-1 \* k \* t) + Tcf  
 timethresh <- log((thresh - Tcf)/(Tci - Tcf))/(-1 \* k)  
 tau <- (rho \* V \* Spheat)/(ATOT \* (hc + hr))  
 dTc <- j - kTc  
 list(Tc = Tc, Tcf = Tcf, tau = tau, dTc = dTc, abs2 = abs2)  
}

##### 

### DEB.R.

DEB.R function available on [**Github**](https://github.com/darwinanddavis/MalishevBullKearney/blob/master/DEB.R).

DEB<-function (step = 1/24, z = 7.997, del\_M = 0.242, F\_m = 13290 \*   
 step, kap\_X = 0.85, v = 0.065 \* step, kap = 0.886, p\_M = 32 \*   
 step, E\_G = 7767, kap\_R = 0.95, k\_J = 0.002 \* step, E\_Hb = 73590,   
 E\_Hj = E\_Hb, E\_Hp = 186500, h\_a = 2.16e-11/(step^2), s\_G = 0.01,   
 T\_REF = 20, TA = 8085, TAL = 18721, TAH = 9E+4, TL = 288,   
 TH = 315, E\_0 = 1040000, f = 1, E\_sm = 1116, K = 1, andens\_deb = 1,   
 d\_V = 0.3, d\_E = 0.3, d\_Egg = 0.3, mu\_X = 525000, mu\_E = 585000,   
 mu\_V = 5e+05, mu\_P = 480000, kap\_X\_P = 0.1, n\_X = c(1, 1.8,   
 0.5, 0.15), n\_E = c(1, 1.8, 0.5, 0.15), n\_V = c(1, 1.8,   
 0.5, 0.15), n\_P = c(1, 1.8, 0.5, 0.15), n\_M\_nitro = c(1,   
 4/5, 3/5, 4/5), clutchsize = 2, clutch\_ab = c(0.085,   
 0.7), viviparous = 0, minclutch = 0, batch = 1, lambda = 1/2,   
 VTMIN = 26, VTMAX = 39, ma = 1e-04, mi = 0, mh = 0.5, arrhenius = matrix(data = matrix(data = c(rep(TA,   
 8), rep(TAL, 8), rep(TAH, 8), rep(TL, 8), rep(TH, 8)),   
 nrow = 8, ncol = 5), nrow = 8, ncol = 5), acthr = 1,   
 X = 10, E\_pres = 6011.93, V\_pres = 3.9752^3, E\_H\_pres = 73592,   
 q\_pres = 0, hs\_pres = 0, surviv\_pres = 1, Es\_pres = 0, cumrepro = 0,   
 cumbatch = 0, p\_B\_past = 0, stage = 1, breeding = 0, pregnant = 0,   
 Tb = 33)   
{  
 q\_init <- q\_pres  
 E\_H\_init <- E\_H\_pres  
 hs\_init <- hs\_pres  
 fecundity <- 0  
 clutches <- 0  
 clutchenergy = E\_0 \* clutchsize  
 n\_O <- cbind(n\_X, n\_V, n\_E, n\_P)  
 CHON <- c(12, 1, 16, 14)  
 wO <- CHON %\*% n\_O  
 w\_V = wO[3]  
 M\_V <- d\_V/w\_V  
 y\_EX<-kap\_X\*mu\_X/mu\_E # yield of reserve on food  
 y\_XE<-1/y\_EX # yield of food on reserve  
 y\_VE<-mu\_E\*M\_V/E\_G # yield of structure on reserve  
 y\_PX<-kap\_X\_P\*mu\_X/mu\_P # yield of faeces on food  
 y\_PE<-y\_PX/y\_EX # yield of faeces on reserve 0.143382353  
 nM <- matrix(c(1, 0, 2, 0, 0, 2, 1, 0, 0, 0, 2, 0, n\_M\_nitro),   
 nrow = 4)  
 n\_M\_nitro\_inv <- c(-1 \* n\_M\_nitro[1]/n\_M\_nitro[4], (-1 \*   
 n\_M\_nitro[2])/(2 \* n\_M\_nitro[4]), (4 \* n\_M\_nitro[1] +   
 n\_M\_nitro[2] - 2 \* n\_M\_nitro[3])/(4 \* n\_M\_nitro[4]),   
 1/n\_M\_nitro[4])  
 n\_M\_inv <- matrix(c(1, 0, -1, 0, 0, 1/2, -1/4, 0, 0, 0, 1/2,   
 0, n\_M\_nitro\_inv), nrow = 4)  
 JM\_JO <- -1 \* n\_M\_inv %\*% n\_O  
 etaO <- matrix(c(y\_XE/mu\_E \* -1, 0, 1/mu\_E, y\_PE/mu\_E, 0,   
 0, -1/mu\_E, 0, 0, y\_VE/mu\_E, -1/mu\_E, 0), nrow = 4)  
 w\_N <- CHON %\*% n\_M\_nitro  
 Tcorr = exp(TA \* (1/(273 + T\_REF) - 1/(273 + Tb)))/(1 + exp(TAL \*   
 (1/(273 + Tb) - 1/TL)) + exp(TAH \* (1/TH - 1/(273 + Tb))))  
 M\_V = d\_V/w\_V  
 p\_MT = p\_M \* Tcorr  
 k\_Mdot = p\_MT/E\_G  
 k\_JT = k\_J \* Tcorr  
 p\_AmT = p\_MT \* z/kap  
 vT = v \* Tcorr  
 E\_m = p\_AmT/vT  
 F\_mT = F\_m \* Tcorr  
 g = E\_G/(kap \* E\_m)  
 E\_scaled = E\_pres/E\_m  
 V\_max = (kap \* p\_AmT/p\_MT)^(3)  
 h\_aT = h\_a \* Tcorr  
 L\_T = 0  
 L\_pres = V\_pres^(1/3)  
 L\_max = V\_max^(1/3)  
 scaled\_l = L\_pres/L\_max  
 kappa\_G = (d\_V \* mu\_V)/(w\_V \* E\_G)  
 yEX = kap\_X \* mu\_X/mu\_E  
 yXE = 1/yEX  
 yPX = kap\_X\_P \* mu\_X/mu\_P  
 mu\_AX = mu\_E/yXE  
 eta\_PA = yPX/mu\_AX  
 w\_X = wO[1]  
 w\_E = wO[3]  
 w\_V = wO[2]  
 w\_P = wO[4]  
 if (E\_H\_pres <= E\_Hb) {  
 dLdt = (vT \* E\_scaled - k\_Mdot \* g \* V\_pres^(1/3))/(3 \*   
 (E\_scaled + g))  
 V\_temp = (V\_pres^(1/3) + dLdt)^3  
 dVdt = V\_temp - V\_pres  
 rdot = vT \* (E\_scaled/L\_pres - (1 + L\_T/L\_pres)/L\_max)/(E\_scaled +   
 g)  
 }  
 else {  
 rdot = vT \* (E\_scaled/L\_pres - (1 + L\_T/L\_pres)/L\_max)/(E\_scaled +   
 g)  
 dVdt = V\_pres \* rdot  
 if (dVdt < 0) {  
 dVdt = 0  
 }  
 }  
 V = V\_pres + dVdt  
 if (V < 0) {  
 V = 0  
 }  
 svl = V^(0.3333333333333)/del\_M \* 10  
 if (E\_H\_pres <= E\_Hb) {  
 Sc = L\_pres^2 \* (g \* E\_scaled)/(g + E\_scaled) \* (1 +   
 ((k\_Mdot \* L\_pres)/vT))  
 dUEdt = -1 \* Sc  
 E\_temp = ((E\_pres \* V\_pres/p\_AmT) + dUEdt) \* p\_AmT/(V\_pres +   
 dVdt)  
 dEdt = E\_temp - E\_pres  
 }  
 else {  
 if (Es\_pres > 1e-07 \* E\_sm \* V\_pres) {  
 dEdt = (p\_AmT \* f - E\_pres \* vT)/L\_pres  
 }  
 else {  
 dEdt = (p\_AmT \* 0 - E\_pres \* vT)/L\_pres  
 }  
 }  
 E = E\_pres + dEdt  
 if (E < 0) {  
 E = 0  
 }  
 p\_M = p\_MT \* V\_pres  
 p\_J = k\_JT \* E\_H\_pres  
 if (Es\_pres > 1e-07 \* E\_sm \* V\_pres) {  
 p\_A = V\_pres^(2/3) \* p\_AmT \* f  
 }  
 else {  
 p\_A = 0  
 }  
 p\_X = p\_A/kap\_X  
 p\_C = (E\_m \* (vT/L\_pres + k\_Mdot \* (1 + L\_T/L\_pres)) \* (E\_scaled \*   
 g)/(E\_scaled + g)) \* V\_pres  
 p\_R = (1 - kap) \* p\_C - p\_J  
 if (E\_H\_pres < E\_Hp) {  
 if (E\_H\_pres <= E\_Hb) {  
 U\_H\_pres = E\_H\_pres/p\_AmT  
 dUHdt = (1 - kap) \* Sc - k\_JT \* U\_H\_pres  
 dE\_Hdt = dUHdt \* p\_AmT  
 }  
 else {  
 dE\_Hdt = (1 - kap) \* p\_C - p\_J  
 }  
 }  
 else {  
 dE\_Hdt = 0  
 }  
 E\_H = E\_H\_init + dE\_Hdt  
 if (E\_H\_pres >= E\_Hp) {  
 p\_D = p\_M + p\_J + (1 - kap\_R) \* p\_R  
 }  
 else {  
 p\_D = p\_M + p\_J + p\_R  
 }  
 p\_G = p\_C - p\_M - p\_J - p\_R  
 if ((E\_H\_pres <= E\_Hp) | (pregnant == 1)) {  
 p\_B = 0  
 }  
 else {  
 if (batch == 1) {  
 batchprep = (kap\_R/lambda) \* ((1 - kap) \* (E\_m \*   
 (vT \* V\_pres^(2/3) + k\_Mdot \* V\_pres)/(1 + (1/g))) -   
 p\_J)  
 if (breeding == 0) {  
 p\_B = 0  
 }  
 else {  
 if (cumrepro < batchprep) {  
 p\_B = p\_R  
 }  
 else {  
 p\_B = batchprep  
 }  
 }  
 }  
 else {  
 p\_B = p\_R  
 }  
 }  
 if (E\_H\_pres > E\_Hp) {  
 if (cumrepro < 0) {  
 cumrepro = 0  
 }  
 else {  
 cumrepro = cumrepro + p\_R \* kap\_R - p\_B\_past  
 }  
 }  
 cumbatch = cumbatch + p\_B  
 if (stage == 2) {  
 if (cumbatch < 0.1 \* clutchenergy) {  
 stage = 3  
 }  
 }  
 if (E\_H <= E\_Hb) {  
 stage = 0  
 }  
 else {  
 if (E\_H < E\_Hj) {  
 stage = 1  
 }  
 else {  
 if (E\_H < E\_Hp) {  
 stage = 2  
 }  
 else {  
 stage = 3  
 }  
 }  
 }  
 if (cumbatch > 0) {  
 if (E\_H > E\_Hp) {  
 stage = 4  
 }  
 else {  
 stage = stage  
 }  
 }  
 if ((cumbatch > clutchenergy) | (pregnant == 1)) {  
 if (viviparous == 1) {  
 if ((pregnant == 0) & (breeding == 1)) {  
 v\_baby = v\_init\_baby  
 e\_baby = e\_init\_baby  
 EH\_baby = 0  
 pregnant = 1  
 testclutch = floor(cumbatch/E\_0)  
 if (testclutch > clutchsize) {  
 clutchsize = testclutch  
 clutchenergy = E\_0 \* clutchsize  
 }  
 if (cumbatch < clutchenergy) {  
 cumrepro\_temp = cumrepro  
 cumrepro = cumrepro + cumbatch - clutchenergy  
 cumbatch = cumbatch + cumrepro\_temp - cumrepro  
 }  
 }  
 if (hour == 1) {  
 v\_baby = v\_baby\_init  
 e\_baby = e\_baby\_init  
 EH\_baby = EH\_baby\_init  
 }  
 if (EH\_baby > E\_Hb) {  
 if ((Tb < VTMIN) | (Tb > VTMAX)) {  
 }  
 cumbatch(hour) = cumbatch(hour) - clutchenergy  
 repro(hour) = 1  
 pregnant = 0  
 v\_baby = v\_init\_baby  
 e\_baby = e\_init\_baby  
 EH\_baby = 0  
 newclutch = clutchsize  
 fecundity = clutchsize  
 clutches = 1  
 pregnant = 0  
 }  
 }  
 else {  
 if ((Tb < VTMIN) | (Tb > VTMAX)) {  
 }  
 if ((Tb < VTMIN) | (Tb > VTMAX)) {  
 }  
 testclutch = floor(cumbatch/E\_0)  
 if (testclutch > clutchsize) {  
 clutchsize = testclutch  
 }  
 cumbatch = cumbatch - clutchenergy  
 repro = 1  
 fecundity = clutchsize  
 clutches = 1  
 }  
 }  
 if (E\_H\_pres > E\_Hb) {  
 if (acthr > 0) {  
 dEsdt = F\_mT \* (X/(K + X)) \* V\_pres^(2/3) \* f - 1 \*   
 (p\_AmT/kap\_X) \* V\_pres^(2/3)  
 }  
 else {  
 dEsdt = -1 \* (p\_AmT/kap\_X) \* V\_pres^(2/3)  
 }  
 }  
 else {  
 dEsdt = -1 \* (p\_AmT/kap\_X) \* V\_pres^(2/3)  
 }  
 if (V\_pres == 0) {  
 dEsdt = 0  
 }  
 Es = Es\_pres + dEsdt  
 if (Es < 0) {  
 Es = 0  
 }  
 if (Es > E\_sm \* V\_pres) {  
 Es = E\_sm \* V\_pres  
 }  
 gutfull = Es/(E\_sm \* V\_pres)  
 if (gutfull > 1) {  
 gutfull = 1  
 }  
 JOJx = p\_A \* etaO[1, 1] + p\_D \* etaO[1, 2] + p\_G \* etaO[1,   
 3]  
 JOJv = p\_A \* etaO[2, 1] + p\_D \* etaO[2, 2] + p\_G \* etaO[2,   
 3]  
 JOJe = p\_A \* etaO[3, 1] + p\_D \* etaO[3, 2] + p\_G \* etaO[3,   
 3]  
 JOJp = p\_A \* etaO[4, 1] + p\_D \* etaO[4, 2] + p\_G \* etaO[4,   
 3]  
 JOJx\_GM = p\_D \* etaO[1, 2] + p\_G \* etaO[1, 3]  
 JOJv\_GM = p\_D \* etaO[2, 2] + p\_G \* etaO[2, 3]  
 JOJe\_GM = p\_D \* etaO[3, 2] + p\_G \* etaO[3, 3]  
 JOJp\_GM = p\_D \* etaO[4, 2] + p\_G \* etaO[4, 3]  
 JMCO2 = JOJx \* JM\_JO[1, 1] + JOJv \* JM\_JO[1, 2] + JOJe \*   
 JM\_JO[1, 3] + JOJp \* JM\_JO[1, 4]  
 JMH2O = JOJx \* JM\_JO[2, 1] + JOJv \* JM\_JO[2, 2] + JOJe \*   
 JM\_JO[2, 3] + JOJp \* JM\_JO[2, 4]  
 JMO2 = JOJx \* JM\_JO[3, 1] + JOJv \* JM\_JO[3, 2] + JOJe \* JM\_JO[3,   
 3] + JOJp \* JM\_JO[3, 4]  
 JMNWASTE = JOJx \* JM\_JO[4, 1] + JOJv \* JM\_JO[4, 2] + JOJe \*   
 JM\_JO[4, 3] + JOJp \* JM\_JO[4, 4]  
 JMCO2\_GM = JOJx\_GM \* JM\_JO[1, 1] + JOJv\_GM \* JM\_JO[1, 2] +   
 JOJe\_GM \* JM\_JO[1, 3] + JOJp\_GM \* JM\_JO[1, 4]  
 JMH2O\_GM = JOJx\_GM \* JM\_JO[2, 1] + JOJv\_GM \* JM\_JO[2, 2] +   
 JOJe\_GM \* JM\_JO[2, 3] + JOJp\_GM \* JM\_JO[2, 4]  
 JMO2\_GM = JOJx\_GM \* JM\_JO[3, 1] + JOJv\_GM \* JM\_JO[3, 2] +   
 JOJe\_GM \* JM\_JO[3, 3] + JOJp\_GM \* JM\_JO[3, 4]  
 JMNWASTE\_GM = JOJx\_GM \* JM\_JO[4, 1] + JOJv\_GM \* JM\_JO[4,   
 2] + JOJe\_GM \* JM\_JO[4, 3] + JOJp\_GM \* JM\_JO[4, 4]  
 O2FLUX = -1 \* JMO2/(T\_REF/Tb/24.4) \* 1000  
 CO2FLUX = JMCO2/(T\_REF/Tb/24.4) \* 1000  
 MLO2 = (-1 \* JMO2 \* (0.082058 \* (Tb + 273.15))/(0.082058 \*   
 293.15)) \* 24.06 \* 1000  
 GH2OMET = JMH2O \* 18.01528  
 #metabolic heat production (Watts) = growth overhead plus dissipation power (maintenance, maturity maintenance,  
 #maturation/repro overheads) plus assimilation overheads. correct to 20 degrees so it can be temperature corrected  
 #in MET.f for the new guessed Tb  
 DEBQMET = ((1 - kappa\_G) \* p\_G + p\_D + (p\_X - p\_A - p\_A \*   
 mu\_P \* eta\_PA))/3600/Tcorr  
 DRYFOOD = -1 \* JOJx \* w\_X  
 FAECES = JOJp \* w\_P  
 NWASTE = JMNWASTE \* w\_N  
 if (pregnant == 1) {  
 wetgonad = ((cumrepro/mu\_E) \* w\_E)/d\_Egg + ((((v\_baby \*   
 e\_baby)/mu\_E) \* w\_E)/d\_V + v\_baby) \* clutchsize  
 }  
 else {  
 wetgonad = ((cumrepro/mu\_E) \* w\_E)/d\_Egg + ((cumbatch/mu\_E) \*   
 w\_E)/d\_Egg  
 }  
 wetstorage = ((V \* E/mu\_E) \* w\_E)/d\_V  
 wetfood = Es/21525.37/(1 - 0.18)  
 wetmass = V \* andens\_deb + wetgonad + wetstorage + wetfood  
 gutfreemass = V \* andens\_deb + wetgonad + wetstorage  
 potfreemass = V \* andens\_deb + (((V \* E\_m)/mu\_E) \* w\_E)/d\_V  
 dqdt = (q\_pres \* (V\_pres/V\_max) \* s\_G + h\_aT) \* (E\_pres/E\_m) \*   
 ((vT/L\_pres) - rdot) - rdot \* q\_pres  
 if (E\_H\_pres > E\_Hb) {  
 q = q\_init + dqdt  
 }  
 else {  
 q = 0  
 }  
 dhsds = q\_pres - rdot \* hs\_pres  
 if (E\_H\_pres > E\_Hb) {  
 hs = hs\_init + dhsds  
 }  
 else {  
 hs = 0  
 }  
 h\_w = ((h\_aT \* (E\_pres/E\_m) \* vT)/(6 \* V\_pres^(1/3)))^(1/3)  
 dsurvdt = -1 \* surviv\_pres \* hs  
 surviv = surviv\_pres + dsurvdt  
 p\_B\_past = p\_B  
 E\_pres = E  
 V\_pres = V  
 E\_H\_pres = E\_H  
 q\_pres = q  
 hs\_pres = hs  
 suriv\_pres = surviv\_pres  
 Es\_pres = Es  
 deb.names <- c("E\_pres", "V\_pres", "E\_H\_pres", "q\_pres",   
 "hs\_pres", "surviv\_pres", "Es\_pres", "cumrepro", "cumbatch",   
 "p\_B\_past", "O2FLUX", "CO2FLUX", "MLO2", "GH2OMET", "DEBQMET",   
 "DRYFOOD", "FAECES", "NWASTE", "wetgonad", "wetstorage",   
 "wetfood", "wetmass", "gutfreemass", "gutfull", "fecundity",   
 "clutches")  
 results\_deb <- c(E\_pres, V\_pres, E\_H\_pres, q\_pres, hs\_pres,   
 surviv\_pres, Es\_pres, cumrepro, cumbatch, p\_B\_past, O2FLUX,   
 CO2FLUX, MLO2, GH2OMET, DEBQMET, DRYFOOD, FAECES, NWASTE,   
 wetgonad, wetstorage, wetfood, wetmass, gutfreemass,   
 gutfull, fecundity, clutches)  
 names(results\_deb) <- deb.names  
 return(results\_deb)  
}

##### 

# Appendix 1

Netlogo IBM decision making model (.nlogo). Available on [**Github**](https://github.com/darwinanddavis/Sleepy_IBM/blob/master/Sleepy%20IBM_v.6.1.1_two%20strategies.nlogo).

### space and time scales

; Spatial scale: 1500 \* 1500 m  
; 1 patch = 2 m  
; 1 tick = 2 min  
; 1 day = 720 ticks  
; 1 tick = 2 bites possible for small food; 4 bites possible for large food

### interface

; Energy cost of individual  
; =========================  
; Movement-cost: Cost (J) of moving one patch (2 m). Calculated from DEB model.  
; Maintenance-cost: Cost (J) of paying maintenance. Calculated from DEB model.  
  
; Energy gain of individual  
; =================  
; Low-food gain: Energy gain (J) from small food items (Brown 1991).  
; kap\_X: Conversion efficiency of assimilated energy from food (J) (Kooijman 2010).  
  
; Food patch growth  
; =================  
; Large-food-initial: Initial energy level (J) of large food items at setup. Parameterised from literature.  
; Small-food-initial: Initial energy level (J) of small food items at setup. Parameterised from literature.  
  
; Individual attributes  
; ===================  
; Maximum-reserve: Maximum reserve level (J). Appears in 'to setup' and 'to make decision'.  
; Minimum-reserve: Define the critical starvation period. Individuals can survive without food for two hours in this state (reasonable estimate).

### globals

globals  
[  
 in-shade? ; Reports TRUE if turtle is in shade  
 in-food? ; Reports TRUE if turtle is in a food patch  
 min-energy ; Minimum food unit level for individual to lose interest and move away from patch. This eliminates the incentive for individuals to return immediately to the previously visited food patch after vacating it.  
 reserve-level ; Reserve level of individual.  
 min-vision ; Minimum (normal) vision range of individuals (Auburn et al. 2009).  
 max-vision ; Maximum vision range of individuals activated by starvation mode. See 'to starving' procedure (Auburn et al. 2009)  
 ctmincount ; Counter for time spent under min\_T\_b\_  
 feedcount ; Counter for time spent in feeding state.  
 restcount ; Counter for the time spent resting in shade  
 searchcount ; Counter for time spent searching for food.  
 starvecount ; Counter for time spent in starvation state.  
 shadecount ; Counter for time spent searching for shade following a feeding bout.  
 transcount ; Counter for frequency of transitions between any of the three activity states--searching, feeding, resting.  
 zenith ; Zenith angle of sun (update-sun procedure).  
 tempXY ; XY coords for drawing homerange   
 gutfull ; Reports gut level of DEB model  
 movelist ; List of cumulative movement costs  
 fh\_ ; String for working dir to export results  
 ]

### turtles-own

turtles-own  
[  
 activity-state ; Individual is either under a Searching, Feeding, or Resting state for each tick. The transition between the various activity states defines the global behavioural repertoire.  
 energy-gain ; Converted energy gained from food  
 T\_b\_ ; Body temperature (T\_b) of individual (Celsius)  
 T\_b\_basking\_ ; Basking body temperature of individual (Celsius)  
 T\_opt\_range ; Foraging body temperature range of individual (Celsius)  
 T\_opt ; Median foraging body temperature of individual (Celsius)  
 T\_opt\_lower\_ ; Lower foraging body temperature of individual (Celsius)  
 T\_opt\_upper\_ ; Upper foraging body temperature of individual (Celsius)  
 min-T\_b\_ ; Lower critical body temperature (min-T\_b) of individual (Celsius)  
 max-T\_b\_ ; Upper critical body temperature (max-T\_b) of individual (Celsius)  
 vision-range ; Vision (no. of patches) range of individual.  
 has-been-starving? ; Results reporter only variable for reporting stavation time only if individual has starved  
 has-been-feeding? ; Results reporter only variable for reporting feeding time only if individual has been feeding  
 X ; List of x coords for homerange  
 Y ; List of y coords for homerange  
 gutthresh\_ ; Threshold for gutlevel to motivate turtle to move  
 V\_pres\_ ; DEB structural volume  
 wetgonad\_ ; DEB wet mass reproductive organ volume  
 wetstorage\_ ; DEB wet mass storage  
 wetfood\_ ; DEB converted food mass  
 ]

### patches-own

patches-own  
[  
 patch-type ; Defines type of patches in environment as either Food or Shade.  
 food-level ; \*> Interface <\* Defines the initial and updated level of energy (J) in food patches. Food level increases (plant growth; see 'Food patch growth' in Interface) and decreases (feeding by individual) with each tick.  
 shade-level ; \*> Interface <\* Defines the initial and updated level of shade in shade patches. Shade levels remain constant throughout simulation.  
 ]

### breeds

breed  
[homeranges homerange]

### setup

to setup  
 ca  
 if Food-patches + Shade-patches > count patches  
 [ user-message (word "Lower the sum of shade and food patches to < " count patches ".")  
 stop ]  
 random-seed 1 ; Outcomment to generate seed for spatial configuration of all patches in the landscape (food and shade). For reproducibility. NB: turtle movement is still stochastic. See below random-seed primitive for complete function.  
 set min-energy Small-food-initial  
 set min-vision 5 ; 10m (Auburn et al. 2009)  
  
 ask patches  
 [set patch-type "Sun"  
 set pcolor (random 1 + blue)]  
  
 let NumFoodPatches Food-patches / 10  
 ask n-of NumFoodPatches patches [  
 ask n-of 10 patches in-radius 4 [ ; Sets 10 random food patches within a 5-patch radius of Food-patches  
 let food-amount random 100  
 ifelse food-amount < 50  
 [set food-level (Small-food-initial) + random-float 1 \* 10 ^ -5] ; Makes only one food patch attractive to turtle because turtles love good chow  
 [set food-level (Large-food-initial) + random-float 1 \* 10 ^ -5]  
 set pcolor PatchColor  
 set patch-type "Food"  
 ]  
 ]  
  
ifelse Shade-density = "Random"[ ; chooser for setting Random or Clumped shade patches (similar to food patch arrangement)  
 let NumShadePatches Shade-patches  
 ask n-of NumShadePatches patches [  
 let shade-amount random 100  
 ifelse shade-amount < 50  
 [set shade-level (Low-shade + random-float 1 \* 10 ^ -5) ; Makes only one shade patch attractive to turtle  
 set pcolor black + 2]  
 [set shade-level (High-shade + random-float 1 \* 10 ^ -5)  
 set pcolor black]  
 set patch-type "Shade"  
 ]  
]  
[  
 let NumShadePatches Shade-patches / 10  
 ask n-of NumShadePatches patches [  
 ask n-of 10 patches in-radius 4 [ ; Sets 10 random food patches within a 5-patch radius of Food-patches  
 let shade-amount random 100  
 ifelse shade-amount < 50  
 [set shade-level (Low-shade + random-float 1 \* 10 ^ -5) ; Makes only one shade patch attractive to turtle  
 set pcolor black + 2]  
 [set shade-level (High-shade + random-float 1 \* 10 ^ -5)  
 set pcolor black]  
 set patch-type "Shade"  
 ]  
]  
]; close else shade loop  
  
ask patch 0 0 [set patch-type "Shade"  
 set pcolor black]  
  
set movelist (list 0)  
; ask one-of patches with [patch-type = "Shade"]  
; [sprout 1]  
crt 1  
  
 random-seed new-seed ; Outcomment to generate seed for spatial configuration of all patches in the landscape (food and shade).  
  
 ask turtle 0  
 [  
 setxy 0 0 ;random-xcor random-ycor  
 set reserve-level Maximum-reserve  
 set T\_b\_basking\_ 14  
 set T\_opt\_range (list 26 27 28 29 30 31 32 33 34 35 ) ; From Pamula thesis  
 set T\_opt\_upper last T\_opt\_range  
 set T\_opt\_lower first T\_opt\_range  
 set T\_opt median T\_opt\_range  
 set min-T\_b\_ min-T\_b  
 set max-T\_b\_ max-T\_b  
 set V\_pres\_ V\_pres  
 set wetgonad\_ wetgonad  
 set wetstorage\_ wetstorage  
 set wetfood\_ wetfood  
 set activity-state "S"  
 set vision-range min-vision  
 if [patch-type] of patch-here = "Shade"  
 [set in-shade? TRUE]  
 if [patch-type] of patch-here = "Food"  
 [set in-food? TRUE]  
 set shape "lizard"  
 set size 2  
 set color red  
 pen-down  
 set X (list xcor)  
 set Y (list ycor)  
 ]  
setup-spatial-plot  
set fh\_ fh  
reset-ticks  
end

### go

to go  
 tick  
 if not any? turtles  
 [  
 get-homerange  
 print "All turtles dead. Check output of model results."  
 repeat 3 [beep wait 0.2]  
 stop  
 save-world  
 ]  
 if (ticks \* 2 / 60 / 24) = No.-of-days  
 [  
 ask turtle 0  
 [report-results]  
 stop  
 save-world  
 ]  
 ifelse show-plots?  
 []  
 [clear-all-plots]  
 ask turtle 0  
 [  
 report-patch-type  
 ask turtles with [reserve-level > Minimum-reserve]  
 [set vision-range min-vision]  
 update-T\_b  
 make-decision  
 set X lput xcor X ; populate X list with turtle X coords to generate home range  
 set Y lput ycor Y ; populate Y list with turtle Y coords to generate home range  
 ]  
 if any? turtles with [reserve-level <= 0]  
 [ask turtle 0 [report-results]  
 stop  
 ]  
 ask patches with [patch-type = "Food"]  
 [update-food-levels]  
end

### update *Tb*

to update-T\_b  
 ask turtles with [T\_b >= max-T\_b]  
 [stop]  
 if T\_b <= min-T\_b  
 [set ctmincount ctmincount + 1]  
 if (ctmincount \* 2 / 60) = ctminthresh  
 [stop]  
end

### make-decision

to make-decision  
  
;-------------------------------------------------------------------------------------  
;-------------------------------------Optimising-------------------------------------  
;-------------------------------------------------------------------------------------  
 ifelse (strategy = "Optimising")  
 [; start optimising loop  
 ifelse (T\_b > T\_opt\_upper) or (T\_b < T\_opt\_lower)  
 [  
 ask turtle 0  
 [;set label "Resting"  
 set activity-state "R"  
 if [patch-type] of patch-here != "Shade"  
 [shade-search]  
 if ([patch-type] of patch-here = "Shade") and (T\_b < T\_b\_basking\_)  
 [set in-shade? TRUE]  
 ]  
 if (activity-state = "R") and (T\_b >= T\_b\_basking\_) and (T\_b < T\_opt\_upper) ; Basking behaviour  
 [set in-shade? FALSE  
 ; set transcount transcount + 1 ; Outcomment to include basking behaviour as activity state  
 ; plotxy xcor ycor  
 ]  
 set restcount restcount + 1  
 ]  
 [; else optimising loop  
 if (activity-state = "R")  
 [  
 set restcount restcount + 1  
 ; set label "Resting"  
 if ((T\_b <= T\_opt\_upper) and (T\_b >= T\_opt\_lower)); and reserve-level < search-energy  
 [set transcount transcount + 1  
 plotxy xcor ycor  
 set activity-state "S"]  
 ; [set activity-state "R"]  
 ]  
  
   
 if (activity-state = "F");   
 [   
 ifelse (gutfull < gutthresh) ;and ([patch-type] of patch-here = "Food") ; if gut is not full, keep feeding  
 [  
 ask turtle 0  
 [handle-food  
 ;set label "Feeding"  
 set has-been-feeding? TRUE]  
 if [patch-type] of patch-here != "Food" ; if patch isn not food, search for food  
 [set activity-state "S"  
 set transcount transcount + 1  
 plotxy xcor ycor  
 set energy-gain 0]  
 if reserve-level >= Maximum-reserve ;   
 [set transcount transcount + 1  
 plotxy xcor ycor  
; ifelse (strategy = "Optimising")  
; [set activity-state "S"]  
 set activity-state "R"  
 stop]  
 ]  
 [;set label "Gut is full" ; otherwise, turtle moves during active hours of the day  
 socialise  
 set searchcount searchcount + 1  
 plotxy xcor ycor  
 ]  
 ]  
  
 if (activity-state = "S")  
 [  
 ask turtle 0  
 [search  
 ; set label "Searching for food"  
 ]  
 set searchcount searchcount + 1  
 if ([patch-type] of patch-here = "Food") and (gutfull < gutthresh)  
 [set transcount transcount + 1  
 plotxy xcor ycor  
 set activity-state "F"]  
 ]  
 ]  
 ]; end optimising loop  
  
  
;-------------------------------------------------------------------------------------  
;-------------------------------------Satisficing-------------------------------------  
;-------------------------------------------------------------------------------------  
  
 [; else satisfice, i.e. move only when gutfull is below the gut threshold  
 ifelse (T\_b > T\_opt\_upper) or (T\_b < T\_opt\_lower) or (gutfull >= gutthresh); 'gutfull' is DEB.R input  
 [  
 ask turtle 0  
 [;set label "Resting"  
 set activity-state "R"  
 ifelse gutfull >= gutthresh and T\_b < T\_opt\_upper and T\_b > T\_opt\_lower  
 [;set label "Full gut"  
 stop ]  
 [if [patch-type] of patch-here != "Shade"  
 [shade-search]]  
 if ([patch-type] of patch-here = "Shade") and (T\_b < T\_b\_basking\_)  
 [set in-shade? TRUE]  
 ]  
 if (activity-state = "R") and (T\_b >= T\_b\_basking\_) and (T\_b < T\_opt\_upper) ; Basking behaviour  
 [set in-shade? FALSE  
 ; set transcount transcount + 1 ; Outcomment to include basking behaviour as activity state  
 ; plotxy xcor ycor  
 ]  
 set restcount restcount + 1  
 ]  
  
 [  
 if (activity-state = "R")  
 [  
 set restcount restcount + 1  
 ; set label "Resting"  
 if ((T\_b <= T\_opt\_upper) and (T\_b >= T\_opt\_lower)); and reserve-level < search-energy  
 [set transcount transcount + 1  
 plotxy xcor ycor  
 set activity-state "S"]  
 ; [set activity-state "R"]  
 ]  
  
   
 if (activity-state = "F")  
 [   
 ifelse (gutfull < gutthresh) ;and ([patch-type] of patch-here = "Food") ; if gut is not full, keep feeding, else stop.  
 [  
 ask turtle 0  
 [handle-food  
 ;set label "Feeding"  
 set has-been-feeding? TRUE]  
 if [patch-type] of patch-here != "Food"  
 [set activity-state "S"  
 set transcount transcount + 1  
 plotxy xcor ycor  
 set energy-gain 0]  
 if reserve-level >= Maximum-reserve ; Turtle will fight between feeding and resting if DEB model not activated i.e. reserve incurs no cost.  
 [set transcount transcount + 1  
 plotxy xcor ycor  
 set activity-state "R"  
 stop]  
 ]  
 [;set label "Gut is full"  
 stop]  
 ]  
  
 if (activity-state = "S")  
 [  
 ask turtle 0  
 [search  
 ; set label "Searching for food"  
 ]  
 set searchcount searchcount + 1  
 if ([patch-type] of patch-here = "Food") ;and ([food-level] of patch-here > min-energy)  
 [set transcount transcount + 1  
 plotxy xcor ycor  
 set activity-state "F"]  
 ]  
 ]  
 ]; end satisficing loop  
end

### search

to search  
 set reserve-level reserve-level - Movement-cost   
 set movelist lput Movement-cost movelist  
 bounce  
 let local-food-patches patches with [(distance myself < [vision-range] of turtle 0) and (patch-type = "Food")]  
 ifelse any? local-food-patches  
 [let my-food-patch local-food-patches with-min [distance myself] ;with-max [food-level]  
 face one-of my-food-patch]  
 [lt random 180 - 90 ]  
 fd 1  
 if [patch-type] of patch-here = "Food"  
 [set activity-state "F"]  
end

### bounce

to bounce  
; Turtles turn a random angle ~180 when encountering a wall  
 ask turtle 0  
 [ if abs pxcor = abs max-pxcor or  
 abs pycor = abs max-pycor  
 [lt random-float 180 ]  
 ]  
end

### handle food

to handle-food  
 set energy-gain Low-food-gain  
 ;set in-food? TRUE  
 set feedcount feedcount + 1  
 set-current-plot "Spatial coordinates of transition between activity states"  
 set-current-plot-pen "Feeding"  
 ifelse [pcolor] of patch-here = 45  
 [set-plot-pen-color 45]  
 [set-plot-pen-color 55]  
 plotxy xcor ycor  
end

### shade search

to shade-search  
 set reserve-level reserve-level - Movement-cost ; add miniscule movement cost to avoid turtle exiting green food patches for one time step when feeding  
 set movelist lput Movement-cost movelist  
 let local-shade-patches patches with [(distance myself < [vision-range] of turtle 0) and (patch-type = "Shade")]  
 ifelse any? local-shade-patches  
 [let my-shade-patch local-shade-patches with-min [distance myself] with-max [shade-level]  
 face one-of my-shade-patch  
 set shadecount shadecount + 1]  
 [lt random 180 - 90]  
 fd 1  
end

### rest

to rest  
 ifelse strategy = "Optimising"  
 [set activity-state "S"]  
 [set activity-state "R"]  
end

### socialise

to socialise  
 set reserve-level reserve-level - Movement-cost ; add miniscule movement cost to avoid turtle exiting green food patches for one time step when feeding  
 set movelist lput Movement-cost movelist  
 bounce  
 lt random 180 - 90  
 fd 1  
 if gutfull < gutthresh  
 [set activity-state "S"]  
end

### update food levels

to update-food-levels  
 let food-deplete food-level - Low-food-gain  
 if (count turtles-here with [activity-state = "F"] > 0) and (gutfull < gutthresh)  
; [ifelse food-level < Large-food-initial  
 [set food-level food-deplete ; yellow food  
 set in-food? TRUE  
 print "In food"]  
; [set food-level food-level - (Low-food-gain \* 2)] ; green food  
; ]  
 if food-level < Small-food-initial  
 [set patch-type "Sun"]  
set pcolor PatchColor  
end

### report patch color

to-report PatchColor  
 let PatColor 0  
 ifelse food-level >= Large-food-initial  
 [set PatColor green]  
 [ifelse food-level >= Small-food-initial  
 [set PatColor yellow]  
 [set PatColor brown]  
 ]  
 report PatColor  
end

### report patch type

to report-patch-type  
ifelse [patch-type] of patch-here = "Food"  
 [set in-food? TRUE]  
 [  
 ifelse [patch-type] of patch-here = "Shade"  
 [set in-shade? TRUE]  
 [set in-shade? FALSE  
 set in-food? FALSE]  
 ]  
end

### report results

to report-results  
 output-print (word "Number of real days:,, " precision (ticks \* 2 / 60 / 24) 5)  
 output-print ""  
 output-print (word "Time spent searching for food (mins/days):, " (searchcount \* 2) " , " precision (searchcount \* 2 / 60 / 24) 3 "")  
 output-print ""  
 output-print (word "Time spent feeding (mins/days):, " (feedcount \* 2) " , " precision (feedcount \* 2 / 60 / 24) 3 "")  
 output-print ""  
 output-print (word "Time spent searching for shade (mins/days):, " (shadecount \* 2) " , " precision (shadecount \* 2 / 60 / 24) 3 "")  
 output-print ""  
 output-print (word "Time spent resting in shade (mins/days):, " (restcount \* 2) " , " precision (restcount \* 2 / 60 / 24) 3 "")  
 output-print ""  
 output-print (word "Time spent in critical starvation (mins/days):, " (starvecount \* 2) " , " precision (starvecount \* 2 / 60 / 24) 3 "")  
 output-print ""  
 output-print (word "Number of transitions between activity states:, " transcount)  
 output-print ""  
 ifelse has-been-feeding? = TRUE  
 [output-print (word "Proportion of feeding to searching:, " precision (feedcount / searchcount) 3)]  
 [output-print (word "Proportion of feeding to searching:, " 0)]  
 output-print ""  
 ifelse has-been-starving? = TRUE  
 [output-print (word "Proportion of feeding to starving:, " precision (feedcount / starvecount) 3)]  
 [output-print (word "Proportion of feeding to starving:, " 0)]  
 output-print ""  
 output-print (word "Patches with pcolor = brown (eaten): " patches with [pcolor = 35])  
 stop;die  
end

### to save world

to save-world ; This procedure saves the model world. The file output procedure then outputs the saved model world as a .txt file to the local dir.  
 let world user-new-file  
 if ( world != false )  
 [  
 file-write world  
 ask patches  
 [  
 file-write pxcor  
 file-write pycor  
 if patch-type = "Food"  
 [file-write pxcor and pycor and (patch-type = "Food") and food-level]  
 if patch-type = "Shade"  
 [file-write pxcor and pycor and (patch-type = "Shade") and shade-level]  
 ]  
 file-close  
 ]  
end

### spatial plot

to setup-spatial-plot  
 set-current-plot "Spatial coordinates of transition between activity states"  
 set-plot-x-range min-pxcor max-pxcor  
 set-plot-y-range min-pycor max-pycor  
 clear-plot  
end

### get home range

to get-homerange  
draw-homerange  
end

### draw home range

to draw-homerange  
 clear-drawing  
 if any? turtles [  
 ask turtle 0  
 [pu  
 hatch-homeranges 1  
 [hide-turtle  
 ; set ID [ID] of myself  
 set color red  
 ]  
 ; draw the homerange  
 foreach tempXY  
 [ask homeranges  
 [move-to patch (item 0 ?) (item 1 ?)  
 pd  
 ]  
 ]  
 ; close the homerange polygon  
 ask homeranges  
 [let lastpoint first tempXY  
 move-to patch (item 0 lastpoint) (item 1 lastpoint)  
 ]  
 ]  
 ]  
end

##### 

# Appendix 2

Energy and heat budget models, including microclimate model (.R). Available on [**Github**](https://github.com/darwinanddavis/MalishevBullKearney).

### Initial setup

# RNL\_new trans model\_with DEB\_1.6.2  
  
# ----------------------------------------------------------------  
# ------------------- initial Mac OS and R config ----------------  
# ----------------------------------------------------------------  
  
#if using Mac OSX Mountain Lion + and not already in JQR, download and open JGR   
# after downloading, load JGR  
install.packages("JGR")  
Sys.setenv(NOAWT=1)  
library(JGR)  
Sys.unsetenv("NOAWT")  
JGR()  
  
# in JGR onwards  
# if already loaded, uninstall RNetlogo and rJava  
p<-c("rJava", "RNetLogo")  
remove.packages(p)  
  
# install Netlogo and rJava from source if haven't already by downloading from CRAN  
# RNetlogo: https://cran.r-project.org/web/packages/RNetLogo/index.html  
# rJava: https://cran.r-project.org/web/packages/rJava/index.html  
dir<- "<directory where RNetlogo and rjava package sources are downloaded>"  
rnl <- "<RNetLogo package file name>" # e.g. "RNetLogo\_1.0-4.tar.gz"   
rj <- "<rJava package file name>" # e.g. "rJava\_0.9-8.tar.gz"   
install.packages(paste0(dir,"/",rnl, repos = NULL, type="source"))  
install.packages(paste0(dir,"/",rj, repos = NULL, type="source"))  
library(RNetLogo); library(rJava)

#### For PC and working Mac OSX

Source DEB.R and onelump\_varenv.R from [**Github**](%22https://github.com/darwinanddavis/MalishevBullKearney%22)

# ------------------- for PC and working Mac OSX ---------------------------  
# ------------------- model setup ---------------------------  
# get packages  
packages <- c("NicheMapR","adehabitatHR","rgeos","sp", "maptools", "raster","rworldmap","rgdal","dplyr")  
install.packages(packages,dependencies = T)  
lapply(packages,library,character.only=T)  
  
#source DEB and heat budget models from https://github.com/darwinanddavis/MalishevBullKearney  
source('DEB.R')  
source('onelump\_varenv.R')  
  
# set dirs  
setwd("<your working dir>") # set wd  
results.path<- "<dir path to store result outputs>" # set results path

### Read in microclimate data

Source metout, soil, shadmet, and shadsoil from [**Github**](%22https://github.com/darwinanddavis/MalishevBullKearney%22)

# read in microclimate data (metout, soil, shadmet, and shadsoil)  
tzone<-paste("Etc/GMT-",10,sep="")  
metout<-read.csv('metout.csv')  
soil<-read.csv('soil.csv')  
shadmet<-read.csv('shadmet.csv')  
shadsoil<-read.csv('shadsoil.csv')  
micro\_sun\_all<-cbind(metout[,2:5],metout[,9],soil[,6],metout[,14:16])  
colnames(micro\_sun\_all)<-c('dates','JULDAY','TIME','TALOC','VLOC','TS','ZEN','SOLR','TSKYC')  
micro\_shd\_all<-cbind(shadmet[,2:5],shadmet[,9],shadsoil[,6],shadmet[,14:16])  
colnames(micro\_shd\_all)<-c('dates','JULDAY','TIME','TALOC','VLOC','TS','ZEN','SOLR','TSKYC')  
  
# choose a day(s) to simulate  
daystart<-paste('09/09/05',sep="") # yy/mm/dd  
dayfin<-paste('10/12/31',sep="") # yy/mm/dd  
micro\_sun<-subset(micro\_sun\_all, format(as.POSIXlt(micro\_sun\_all$dates), "%y/%m/%d")>=daystart & format(as.POSIXlt(micro\_sun\_all$dates), "%y/%m/%d")<=dayfin)  
micro\_shd<-subset(micro\_shd\_all, format(as.POSIXlt(micro\_shd\_all$dates), "%y/%m/%d")>=daystart & format(as.POSIXlt(micro\_shd\_all$dates), "%y/%m/%d")<=dayfin)  
days<-as.numeric(as.POSIXlt(dayfin)-as.POSIXlt(daystart))  
  
# create time vectors  
time<-seq(0,(days+1)\*60\*24,60) #60 minute intervals from microclimate output  
time<-time[-1]  
times2<-seq(0,(days+1)\*60\*24,2) #two minute intervals for prediction  
time<-time\*60 # minutes to seconds  
times2<-times2\*60 # minutes to seconds  
  
# apply interpolation functions  
velfun<- approxfun(time, micro\_sun[,5], rule = 2)  
Zenfun<- approxfun(time, micro\_sun[,7], rule = 2)  
Qsolfun\_sun<- approxfun(time, micro\_sun[,8], rule = 2)  
Tradfun\_sun<- approxfun(time, rowMeans(cbind(micro\_sun[,6],micro\_sun[,9])), rule = 2)  
Tairfun\_sun<- approxfun(time, micro\_sun[,4], rule = 2)  
Qsolfun\_shd<- approxfun(time, micro\_shd[,8]\*.1, rule = 2)  
Tradfun\_shd<- approxfun(time, rowMeans(cbind(micro\_shd[,6],micro\_shd[,9])), rule = 2)  
Tairfun\_shd<- approxfun(time, micro\_shd[,4], rule = 2)  
  
# upper and lower activity thermal limits  
VTMIN<- 26   
VTMAX<- 35

### Read in DEB parameters

Source DEB\_pars\_Tiliqua\_rugosa.csv from [**Github**](%22https://github.com/darwinanddavis/MalishevBullKearney%22)

# \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* read in DEB parameters \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
  
debpars=as.data.frame(read.csv('DEB\_pars\_Tiliqua\_rugosa.csv',header=FALSE))$V1 # read in DEB pars  
  
# set core parameters  
z=debpars[8] # zoom factor (cm)  
F\_m = 13290 # max spec searching rate (l/h.cm^2)  
kap\_X=debpars[11] # digestion efficiency of food to reserve (-)  
v=debpars[13] # energy conductance (cm/h)  
kap=debpars[14] # kappa, fraction of mobilised reserve to growth/maintenance (-)  
kap\_R=debpars[15] # reproduction efficiency (-)  
p\_M=debpars[16] # specific somatic maintenance (J/cm3)  
k\_J=debpars[18] # maturity maint rate coefficient (1/h)  
E\_G=debpars[19] # specific cost for growth (J/cm3)  
E\_Hb=debpars[20] # maturity at birth (J)  
E\_Hp=debpars[21] # maturity at puberty (J)  
h\_a=debpars[22]\*10^-1 # Weibull aging acceleration (1/h^2)  
s\_G=debpars[23] # Gompertz stress coefficient (-)  
  
# set thermal respose curve paramters  
T\_REF = debpars[1]-273  
TA = debpars[2] # Arrhenius temperature (K)  
TAL = debpars[5] # low Arrhenius temperature (K)  
TAH = debpars[6] # high Arrhenius temperature (K)  
TL = debpars[3] # low temp boundary (K)  
TH = debpars[4] # hight temp boundary (K)  
  
# set auxiliary parameters  
del\_M=debpars[9] # shape coefficient (-)   
E\_0=debpars[24] # energy of an egg (J)  
mh = 1 # survivorship of hatchling in first year  
mu\_E = 585000 # molar Gibbs energy (chemical potential) of reserve (J/mol)  
E\_sm=186.03\*6  
gutfull <- 1  
# set initial state  
E\_pres\_init = (debpars[16]\*debpars[8]/debpars[14])/(debpars[13]) # initial reserve  
E\_m <- E\_pres\_init  
E\_H\_init = debpars[21] + 5  
  
#### change inital size here by multiplying by < 0.85 ####  
V\_pres\_init = (debpars[26] ^ 3) \* 0.85   
d\_V<-0.3  
mass <- V\_pres\_init + V\_pres\_init\*E\_pres\_init/mu\_E/d\_V\*23.9  
  
# \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* end TRANSIENT MODEL SETUP \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
# \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

### Initialise decision-making and DEB models

Source Netlogo model from [**Github**](%22https://github.com/darwinanddavis/MalishevBullKearney%22)

# \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* start NETLOGO SIMULATION \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
  
nl.path<- "<dir path to Netlogo program>"   
ver <-"<version number of Netlogo>" # type in version of Netlogo e.g. "6.0.1"  
# if error, try adding "/app" to end of dir path for running in Windows and El Capitan for OSX  
# nl.path<-"<dir path to Netlogo program>/app"   
NLStart(nl.path)  
NLStart(nl.path, nl.jarname = paste0("netlogo-",ver,".jar"))  
model.path<- "<dir path to Netlogo model>"  
NLLoadModel(model.path)  
  
# \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* setup NETLOGO MODEL \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
  
# 1. update animal and env traits  
month<-"sep"  
NL\_days<-117 # No. of days simulated  
NL\_gutthresh<-0.75  
gutfull<-0.8  
  
# set resource density  
if(density=="high"){  
 NL\_shade<-100000L # Shade patches  
 NL\_food<-100000L # Food patches  
 }else{   
 NL\_shade<-1000L # Shade patches  
 NL\_food<-1000L # Food patches  
}  
  
# 2. update initial conditions for DEB model   
Es\_pres\_init<-(E\_sm\*gutfull)\*V\_pres\_init  
acthr<-1  
Tb\_init<-20  
step = 1/24  
debout<-DEB(step = step, z = z, del\_M = del\_M, F\_m = F\_m \*   
 step, kap\_X = kap\_X, v = v \* step, kap = kap, p\_M = p\_M \*   
 step, E\_G = E\_G, kap\_R = kap\_R, k\_J = k\_J \* step, E\_Hb = E\_Hb,   
 E\_Hj = E\_Hb, E\_Hp = E\_Hp, h\_a = h\_a/(step^2), s\_G = s\_G,   
 T\_REF = T\_REF, TA = TA, TAL = TAL, TAH = TAH, TL = TL,   
 TH = TH, E\_0 = E\_0, E\_pres=E\_pres\_init, V\_pres=V\_pres\_init, E\_H\_pres=E\_H\_init, acthr = acthr, breeding = 1, Es\_pres = Es\_pres\_init, E\_sm = E\_sm)  
  
# 3. calc direct movement cost  
V\_pres<-debout[2]  
step<-1/24 #hourly  
  
p\_M2<-p\_M\*step #J/h  
p\_M2<-p\_M2\*V\_pres # loco cost \* structure  
names(p\_M2)<-NULL # remove V\_pres name attribute from p\_M  
  
# movement cost for time period  
VO2<-0.45 # O2/g/h JohnAdler etal 1986  
  
# multiple p\_M by structure = movement cost (diff between p\_M with loco cost and structure for movement period)  
# p\_M with loco cost   
loco<-VO2\*mass\*20.1 # convert ml O2 to J = J/h   
loco<-loco+p\_M2 # add to p\_M = J/h  
loco<-loco/30/V\_pres ; loco #J/cm3/2min  
  
Es\_pres\_init<-(E\_sm\*gutfull)\*V\_pres\_init  
X\_food<-3000  
V\_pres<-debout[2]  
wetgonad<-debout[19]  
wetstorage<-debout[20]  
wetfood<-debout[21]  
ctminthresh<-120000  
Tairfun<-Tairfun\_shd  
Tc\_init<-Tairfun(1)+0.1 # Initial core temperature  
  
NL\_T\_b<-Tc\_init # Initial T\_b  
NL\_T\_b\_min<-VTMIN # Min foraging T\_b  
NL\_T\_b\_max<-VTMAX # Max foraging T\_b  
NL\_ctminthresh<-ctminthresh # No. of consecutive hours below CTmin that leads to death  
NL\_reserve<-E\_m # Initial reserve density  
NL\_max\_reserve<-E\_m # Maximum reserve level  
NL\_maint<-round(p\_M, 3) # Maintenance cost  
NL\_move<-round(loco, 3) # Movement cost  
NL\_zen<-Zenfun(1\*60\*60) # Zenith angle  
  
strategy<-function(strategy){ # set movement strategy   
 if (strategy == "O"){  
 NLCommand("set strategy \"Optimising\" ")   
 }else{  
 NLCommand("set strategy \"Satisficing\" ")   
 }  
 }  
strategy("O") # "S"  
  
shadedens<-function(shadedens){ # set movement strategy   
 if (shadedens == "Random"){  
 NLCommand("set Shade-density \"Random\" ")   
 }else{  
 NLCommand("set Shade-density \"Clumped\" ")   
 }  
 }  
shadedens("Clumped") # set clumped resources

### Run simulation

sc<-1 # set no. of desired simualations---for automating writing of each sim results to file. N = N runs  
for (i in 1:sc){ # start sc sim loop  
  
NLCommand("set Shade-patches",NL\_shade,"set Food-patches",NL\_food,"set No.-of-days",NL\_days,"set T\_b precision",  
NL\_T\_b, "2","set T\_opt\_lower precision", NL\_T\_b\_min, "2","set T\_opt\_upper precision", NL\_T\_b\_max, "2",  
"set reserve-level", NL\_reserve, "set Maximum-reserve", NL\_max\_reserve, "set Maintenance-cost", NL\_maint,  
"set Movement-cost precision", NL\_move, "3", "set zenith", NL\_zen, "set ctminthresh", NL\_ctminthresh,   
"set gutthresh", NL\_gutthresh, 'set gutfull', gutfull, 'set V\_pres precision', V\_pres, "5", 'set wetstorage precision', wetstorage, "5",   
'set wetfood precision', wetfood, "5", 'set wetgonad precision', wetgonad, "5","setup")  
  
#NLCommand("inspect turtle 0")  
  
NL\_ticks<-NL\_days / (2 / 60 / 24) # No. of NL ticks (measurement of days)  
NL\_T\_opt\_l<-NLReport("[T\_opt\_lower] of turtle 0")  
NL\_T\_opt\_u<-NLReport("[T\_opt\_upper] of turtle 0")  
  
# data frame setup for homerange polygon  
turtles<-data.frame() # make an empty data frame  
NLReport("[X] of turtle 0"); NLReport("[Y] of turtle 0")  
who<-NLReport("[who] of turtle 0")  
  
# \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
# \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* start NETLOGO SIMULATION \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
  
debcall<-0 # check for first call to DEB  
stepcount<-0 # DEB model step count  
  
for (i in 1:NL\_ticks){  
stepcount<-stepcount+1  
NLDoCommand(1, "go")  
  
######### Reporting presence of shade  
shade<-NLGetAgentSet("in-shade?","turtles", as.data.frame=T); shade<-as.numeric(shade) # returns an agentset of whether turtle is currently on shade patch  
  
# choose sun or shade  
tick<-i  
times3<-c(times2[tick],times2[tick+1])  
  
if(shade==0){  
 Qsolfun<-Qsolfun\_sun  
 Tradfun<-Tradfun\_sun  
 Tairfun<-Tairfun\_sun  
}else{  
 Qsolfun<-Qsolfun\_shd  
 Tradfun<-Tradfun\_shd  
 Tairfun<-Tairfun\_shd  
}  
if(i==1){  
Tc\_init<-Tairfun(1)+0.1 #initial core temperature  
}  
  
# one\_lump\_trans params  
Qsol<-Qsolfun(mean(times3)); Qsol  
vel<-velfun(mean(times3)) ;vel  
Tair<-Tairfun(mean(times3));Tair  
Trad<-Tradfun(mean(times3)); Trad  
Zen<-Zenfun(mean(times3)); Zen  
  
# calc Tb params at 2 mins interval  
Tbs<-onelump\_varenv(t=120,time=times3[2],Tc\_init=Tc\_init,thresh = 30, AMASS = mass, lometry = 3, Tairf=Tairfun,Tradf=Tradfun,velf=velfun,Qsolf=Qsolfun,Zenf=Zenfun)  
Tb<-Tbs$Tc  
rate<-Tbs$dTc  
Tc\_init<-Tb  
  
NLCommand("set T\_b precision", Tb, "2") # Updating Tb  
NLCommand("set zenith", Zenfun(times3[2])) # Updating zenith  
  
# time spent below VTMIN  
ctminhours<-NLReport("[ctmincount] of turtle 0") \* 2/60 # ticks to hours  
if (ctminhours == NL\_ctminthresh) {NLCommand("ask turtle 0 [stop]")}  
  
# \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* start DEB SIMULATION \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
  
if(stepcount==1) { # run DEB loop every time step (2 mins)  
stepcount<-0  
  
# report activity state  
actstate<-NLReport("[activity-state] of turtle 0")  
 # Reports true if turtle is in food   
actfeed<-NLGetAgentSet("in-food?","turtles", as.data.frame=T); actfeed<-as.numeric(actfeed)  
   
n<-1 # time steps  
step<-2/1440 # step size (2 mins). For hourly: 1/24  
# update direct movement cost  
if(actstate == "S"){  
 NLCommand("set Movement-cost", NL\_move)  
 }else{  
 NLCommand("set Movement-cost", 1e-09)  
 }   
# if within activity range, it's daytime, and gut below threshold   
if(Tbs$Tc>=VTMIN & Tbs$Tc<=VTMAX & Zen!=90 & gutfull<=NL\_gutthresh){   
 acthr=1 # activity state = 1   
if(actfeed==1){ # if in food patch  
 X\_food<-NLReport("[energy-gain] of turtle 0") # report joules intake  
 }  
 }else{  
 X\_food = 0   
 acthr=0  
 }  
  
# calculate DEB output   
if(debcall==0){  
 # initialise DEB  
 debout<-matrix(data = 0, nrow = n, ncol = 26)  
 deb.names<-c("E\_pres","V\_pres","E\_H\_pres","q\_pres","hs\_pres","surviv\_pres","Es\_pres","cumrepro","cumbatch","p\_B\_past","O2FLUX","CO2FLUX","MLO2","GH2OMET","DEBQMET","DRYFOOD","FAECES","NWASTE","wetgonad","wetstorage","wetfood","wetmass","gutfreemass","gutfull","fecundity","clutches")  
 colnames(debout)<-deb.names  
 # initial conditions  
 debout<-DEB(E\_pres=E\_pres\_init, V\_pres=V\_pres\_init, E\_H\_pres=E\_H\_init, acthr = acthr, Tb = Tb\_init, breeding = 1, Es\_pres = Es\_pres\_init, E\_sm = E\_sm, step = step, z, del\_M = del\_M, F\_m = F\_m \*   
 step, kap\_X = kap\_X, v = v \* step, kap = kap, p\_M = p\_M \*   
 step, E\_G = E\_G, kap\_R = kap\_R, k\_J = k\_J \* step, E\_Hb = E\_Hb,   
 E\_Hj = E\_Hb, E\_Hp = E\_Hp, h\_a = h\_a/(step^2), s\_G = s\_G,   
 T\_REF = T\_REF, TA = TA, TAL = TAL, TAH = TAH, TL = TL,   
 TH = TH, E\_0 = E\_0)  
 debcall<-1  
 }else{  
 debout<-DEB(step = step, z = z, del\_M = del\_M, F\_m = F\_m \*   
 step, kap\_X = kap\_X, v = v \* step, kap = kap, p\_M = p\_M \*   
 step, E\_G = E\_G, kap\_R = kap\_R, k\_J = k\_J \* step, E\_Hb = E\_Hb,   
 E\_Hj = E\_Hb, E\_Hp = E\_Hp, h\_a = h\_a/(step^2), s\_G = s\_G,   
 T\_REF = T\_REF, TA = TA, TAL = TAL, TAH = TAH, TL = TL,   
 TH = TH, E\_0 = E\_0,   
 X=X\_food,acthr = acthr, Tb = Tbs$Tc, breeding = 1, E\_sm = E\_sm, E\_pres=debout[1],V\_pres=debout[2],E\_H\_pres=debout[3],q\_pres=debout[4],hs\_pres=debout[5],surviv\_pres=debout[6],Es\_pres=debout[7],cumrepro=debout[8],cumbatch=debout[9],p\_B\_past=debout[10])  
 }  
mass<-debout[22]  
gutfull<-debout[24]  
NL\_reserve<-debout[1]  
V\_pres<-debout[2]  
wetgonad<-debout[19]  
wetstorage<-debout[20]  
wetfood<-debout[21]   
  
#update NL wetmass properties   
NLCommand("set V\_pres precision", V\_pres, "5")  
NLDoCommand("plot xcor ycor")  
NLCommand("set wetgonad precision", wetgonad, "5")  
NLDoCommand("plot xcor ycor")  
NLCommand("set wetstorage precision", wetstorage, "5")  
NLDoCommand("plot xcor ycor")  
NLCommand("set wetfood precision", wetfood, "5")  
NLDoCommand("plot xcor ycor")   
   
  
} #--- end DEB loop  
  
NLCommand("set reserve-level", NL\_reserve) # update reserve  
NLCommand("set gutfull", debout[24])# update gut level  
  
# \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* end DEB SIMULATION \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
  
# generate results, with V\_pres, wetgonad, wetstorage, and wetfood from debout  
if(i==1){  
 results<-cbind(tick,Tb,rate,shade,V\_pres,wetgonad,wetstorage,wetfood,NL\_reserve)   
 }else{  
 results<-rbind(results,c(tick,Tb,rate,shade,V\_pres,wetgonad,wetstorage,wetfood,NL\_reserve))  
 }  
results<-as.data.frame(results)  
  
# generate data frames for homerange polygon  
if (tick == NL\_ticks - 1){  
 X<-NLReport("[X] of turtle 0"); head(X)  
 Y<-NLReport("[Y] of turtle 0"); head(Y)  
 turtles<-data.frame(X,Y)  
 who1<-rep(who,NL\_ticks); who # who1<-rep(who,NL\_ticks - 1); who   
 turtledays<-rep(1:NL\_days,length.out=NL\_ticks,each=720)   
 turtle<-data.frame(ID = who1,days=turtledays)  
 turtles<-cbind(turtles,turtle)  
 }  
  
} # \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* end NL loop \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
  
# get hr data  
spdf<-SpatialPointsDataFrame(turtles[1:2], turtles[3]) # creates a spatial points data frame (adehabitatHR package)  
homerange<-mcp(spdf,percent=95)  
  
# writing new results  
if (exists("results")){ #if results exist  
 sc<-sc-1   
 nam <- paste("results", sc, sep = "") # generate new name with added sc count  
 rass<-assign(nam,results) #assign new name to results. call 'results1, results2 ... resultsN'  
 namh <- paste("turtles", sc, sep = "") #generate new name with added sc count  
 rassh<-assign(namh,turtles) #assign new name to results. call 'results1, results2 ... resultsN'  
 nams <- paste("spdf", sc, sep = "")   
 rasss<-assign(nams,spdf)   
 namhr <- paste("homerange", sc, sep = "")   
 rasshr<-assign(namhr,homerange)   
  
 fh<-results.path; fh  
 for (i in rass){  
 # export all results  
 write.table(results,file=paste(fh,nam,".R",sep=""))  
 }  
 for (i in rassh){  
 # export turtle location data  
 write.table(turtles,file=paste(fh,namh,".R",sep=""))  
 }  
 #export NL plots  
 month<-"sep"  
 #spatial plot  
 sfh<-paste(month,NL\_days,round(mass,0),NL\_shade,as.integer(NL\_food\*10),"\_",sc,"\_move","",sep="");sfh  
 NLCommand(paste("export-plot \"Spatial coordinates of transition between activity states\" \"",results.path,sfh,".csv\"",sep=""))  
 #temp plot   
 tfh<-paste(month,NL\_days,round(mass,0),NL\_shade,as.integer(NL\_food\*10),"\_",sc,"\_temp",sep="")  
 NLCommand(paste("export-plot \"Body temperature (T\_b)\" \"",results.path,tfh,".csv\"",sep=""))  
 #activity budget  
 afh<-paste(month,NL\_days,round(mass,0),NL\_shade,as.integer(NL\_food\*10),"\_",sc,"\_act","",sep="");afh  
 NLCommand(paste("export-plot \"Global time budget\" \"",results.path,afh,".csv\"",sep=""))  
 #text output  
 xfh<-paste(month,NL\_days,round(mass,0),NL\_shade,as.integer(NL\_food\*10),"\_",sc,"\_txt",sep="");xfh  
 NLCommand(paste("export-output \"",results.path,xfh,".csv\"",sep=""))  
 #gut level  
 gfh<-paste(month,NL\_days,round(mass,0),NL\_shade,as.integer(NL\_food\*10),"\_",sc,"\_gut","",sep="");gfh  
 NLCommand(paste("export-plot \"Gutfull\" \"",results.path,gfh,".csv\"",sep=""))  
 #wet mass   
 mfh<-paste(month,NL\_days,round(mass,0),NL\_shade,as.integer(NL\_food\*10),"\_",sc,"\_wetmass","",sep="");mfh  
 NLCommand(paste("export-plot \"Total wetmass plot\" \"",results.path,mfh,".csv\"",sep=""))  
 #movement cost (loco)   
 lfh<-paste(month,NL\_days,round(mass,0),NL\_shade,as.integer(NL\_food\*10),"\_",sc,"\_loco","",sep="");lfh  
 NLCommand(paste("export-plot \"Movement costs\" \"",results.path,lfh,".csv\"",sep=""))  
 }  
} # \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* end sc sim loop \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* end NETLOGO SIMULATION \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

### Example of data output files from simulation

# example files of results output in results.path  
list.files(results.path)  
# [1] "results0.R" "sep572310001000\_0\_act.csv"   
# [3] "sep572310001000\_0\_gut.csv" "sep572310001000\_0\_loco.csv"   
# [5] "sep572310001000\_0\_move.csv" "sep572310001000\_0\_temp.csv"   
# [7] "sep572310001000\_0\_txt.csv" "sep572310001000\_0\_wetmass.csv"  
# [9] "turtles0.R"

##### 

# Appendix 3

### Summary of DEB parameters and primary metabolic pathways.

In DEB theory, flows of energy and mass from food are tracked through time to predict the individual state in terms of growth (structure, , i.e. volume of body tissue built during growth and which requires maintenance), body condition (reserve, , the chemical intermediary between the transformation of food and the growth and maintenance of structure), and maturity (, the energy invested in increasing the maturation state, i.e. energetic costs of development, all of which is dissipated) (Kooijman 2010). Temperature influences all rates according to the Arrhenius model. Food is eaten then converted to reserve, which is subsequently mobilized to fuel the rest of the metabolism. Reserve is readily available pools of generalised compounds, rather than simply energy or fat storage, contained within individual body cells and collectively, per structural volume, measured as reserve density, . Food intake scales with food density following a functional response (Table 1), whereby either searching or handling limits individuals as increases from 0 to 1, respectively. Following an allocation rule ( rule), reserves are mobilised to structure via the natural hierarchy of cell metabolism—first to somatic maintenance then to growth . The remaining fixed fraction of mobilised energy then fuels maturation and reproduction. Animals switch between three maturity stages—embryo, juvenile, and adult—defined by maturity thresholds representing the cumulative energy invested in maturation. Energy is also invested in maintenance of a given level of maturity and, once reaching puberty, excess energy beyond maturity maintenance costs is invested in reproductive biomass . See Fig. 4 in Kearney et al. (2013) for a schematic breakdown of the above processes.

In DEB theory, reserve dynamics drive metabolism. The rate of assimilated food into energy follows the Holling Type II functional response

where {} is the maximum assimilation rate and the volumetric body length . Parameters surrounded by square [\*] and curly {\*} parentheses are per volume and surface area, respectively. Energy is then mobilised from reserves with a constant fraction going to somatic maintenance and somatic growth

where is energy conductance of mobilised reserve and is specific growth rate, i.e. change in structure over time

with the somatic maintenance cost **(Eq. 5)** and the cost of producing structural tissue (both biomass and overhead costs). This relationship means growth dilutes reserve levels, measured as reserve density per volume , as it rises and falls with incoming energy minus energy used for metabolic processes , giving the reserve dynamics equation

*Maintenance*  
Heat, water, and CO2 are expelled as products from the costs of maintaining the volume and surface of structure

Further costs include the overheads of reproduction, feeding (heat increment of ), and growth , including growth overhead costs and tissue biomass

while maturity maintenance from , measured by energy dissipated as heat, contributes to maintaining the current and transitioning to the next maturity level

where is a coefficient controlling the rate of maturity maintenance. The organism first pays maturity maintenance and, in an immature individual, the remaining flux feeds further increases in maturity

*Growth*  
Animals grow in structural length (~ maximum ) with **(Eq. 3)** only after paying maintenance ; maintaining and growing cells incurs growth costs of new biovolume, calculated as an energy investment ratio

where is maximum reserve density when . Reserve density is scaled to to interpret changes in as animals encounter food at different densities over time, giving

so under steady state, i.e. when food is constant, . In the standard DEB model, body shape remains constant (isomorphic) during growth. Therefore, the body surface area is proportional to volume . Following **Eq. 3** so that given , an isomorphic animal increases in body length under constant food following

##### 

# Figures

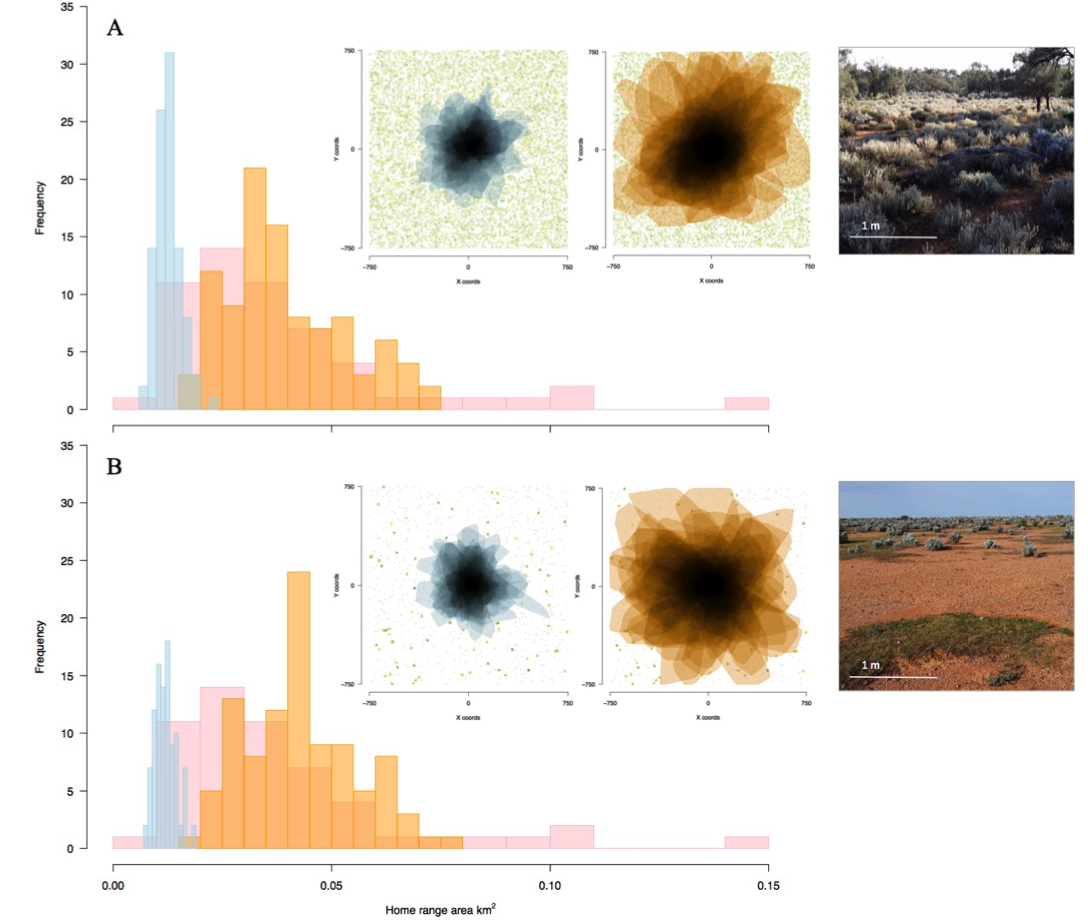


Figure S1. Distributions of home range area (km2) of real animals (pink) and simulated optimising (orange) and satisficing (blue) movement strategies under (A) dense and (B) sparse resource distribution (food and shade). Insets (L–R): Home range polygons in space showing overlap of simulated satisficing (blue) and optimising (orange) movement strategies, and examples of (upper) dense and (lower) sparse resource distributions in the study site.

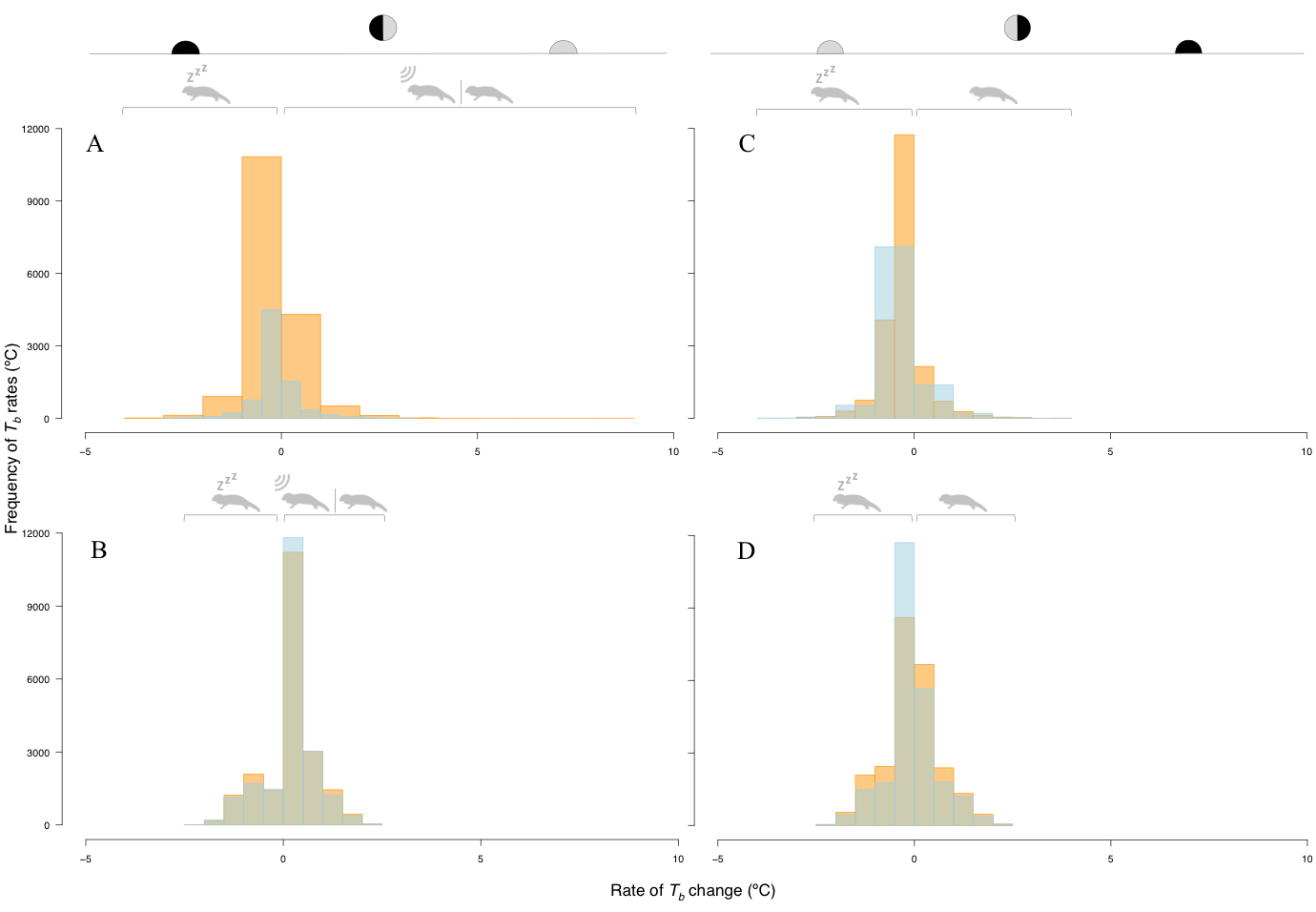


Figure S2. Rates of Tb change (ºC 2 min–1) comparing (A) observed active (orange; #11885) and passive (blue; #11533) movement and (B) simulated optimising (orange) and satisficing (blue) movement for the morning hours (heating period; 06:00–12:00). (C) Observed active (orange) and passive (blue) movement and (D) simulated optimising (orange) and satisficing (blue) movement for the afternoon hours (cooling period; 12:00–18:00) throughout the breeding season. Animal graphics represent the most probable activity state of the animal (from Fig. 2).

##### 

# References

Auburn, Z. M., Bull, C. M. and Kerr G. D. (2009) The visual perceptual range of a lizard, *Tiliqua rugosa*, Journal of Ethology, 27: 75-81.

Brown, G. W. (1991) Ecological feeding analysis of south-eastern Australian Scincids (Reptilia–Lacertilia), Australian Journal of Zoology, 39: 9-29.

Kearney, M. R., Simpson, S. J., Raubenheimer, D. & Kooijman, S. A. L. M. (2012) Balancing heat, water and nutrients under environmental change: a thermodynamic niche framework. Functional Ecology, 27(4): 950–966.

Kooijman, S.A.L.M., (2010). Dynamic Energy Budget Theory, Cambridge, UK: Cambridge University Press.

Pamula, Y. (1997) Ecological and physiological aspects of reproduction in the viviparous skink *Tiliqua rugosa*, Ph.D. thesis, Flinders University, South Australia, Australia.

1. This Supplementary Material can be found at <https://github.com/darwinanddavis/MalishevBullKearney> or <https://doi.org/10.5281/zenodo.998145>. [↑](#footnote-ref-1)