A

Appendix A

Matthew Malishev[1,2*], C. Michael Bull[3], & Michael R. Kearney[2]

[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

This Appendix is found at https://doi.org/10.5281/zenodo.998145 or on Github at https://github.com/darwinanddavis/Thesis/tree/master/AppendixA.

[*]Corresponding author: matthew.malishev@gmail.com

Table of Contents

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.

A

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 ut of these patches at each time step, the animal updates its current T_b, including rates of change in T_b per 2-minute ime step.

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

A

onelump_varenv.R.

onelump_varenv.R available on **Github**.

```
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
```

A

```r
    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
```

A

```
}
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
```

A

```
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)
```

A

```
  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)
}
```

A

DEB.R.

DEB.R function available on **Github**.

```
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)
```

A

```
    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
```

A

```
            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) {
```

A

```
          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]
```

A

```
  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 (mainte
nance, maturity maintenance,
 #maturation/repro overheads) plus assimilation overheads. correct to 20 degrees so it c
an 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)
```

A

```
  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)
}
```

A

Appendix A1

Netlogo IBM decision making model (.nlogo). Available on **Github**.

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 201
0).

; Food patch growth
; ==================
; Large-food-initial: Initial energy level (J) of large food items at setup. Parameterised fro
m literature.
; Small-food-initial: Initial energy level (J) of small food items at setup. Parameterised fro
m literature.

; Individual attributes
; ====================
; Maximum-reserve: Maximum reserve level (J). Appears in 'to setup' and 'to make decisi
on'.
; 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 awa
y from patch. This eliminates the incentive for individuals to return immediately to the pr
eviously 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. S
ee '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 s
tates--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 eac
h tick. The transition between the various activity states defines the global behavioural re
pertoire.
  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 in
dividual has starved
  has-been-feeding?  ; Results reporter only variable for reporting feeding time only if ind
ividual 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 patche
s (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 pat
ches 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
```

A

```
    ]
  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 T_b

```
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-------------------------------------
;-----------------------------------------------------------------------------------
```

A

```
 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) ; Baskin
g 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-ene
rgy
     [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 fu
ll, 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) ; Baskin
g 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-ene
rgy
     [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 fu
ll, 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 i
```

```
f 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-energ
y)
      [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 (p
atch-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
```

A

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-le
vel]
    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 (f
eedcount * 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) " , " pre
cision (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 / searc
hcount) 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 / starvec
ount) 3)]
   [output-print (word "Proportion of feeding to starving:, " 0)]
   output-print ""
   output-print (word "Patches with pcolor = brown (eaten): " patches with [pcolor = 35])
```

A

```
  stop;die
end
```

to save world

```
to save-world ; This procedure saves the model world. The file output procedure then out puts 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
```

A

Appendix A2

Energy and heat budget models, including microclimate model (.R). Available on **Github**.

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
dir<- "<your working directory>"
install.packages(paste0(dir,"/RNetLogo_1.0-2.tar.gz", repos = NULL, type="source"))
install.packages(paste0(dir,"/rJava_0.9-8.tar.gz", repos = NULL, type="source"))
library(RNetlogo); library(rJava)
```

For PC and working Mac OSX

Source DEB.R and onelump_varenv.R from **Github**

```
# ------------------- for PC and working Mac OSX -----------------
# ------------------- model setup ---------------------------
# get packages
install.packages(c("NicheMapR","adehabitatHR","rgeos","sp", "maptools", "raster","rw
orldmap","rgdal","dplyr"))
library(NicheMapR); library(adehabitatHR); library(rgeos); library(sp); library(mapt
ools); library(raster); library(rworldmap); library(rgdal); library(dplyr)

#source DEB and heat budget models from https://github.com/darwinanddavis/Thesis/tre
```

A

```
e/master/AppendixA
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**

```
# 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**

```
# ************************** 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**

```
# ********************* start NETLOGO SIMULATION ********************
**

nl.path<- "<dir path to Netlogo program>"
NLStart(nl.path)
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 struct
ure 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 deat
h
NL_reserve<-E_m       # Initial reserve density
```

A

```
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 simulations---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-d
ays",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 Mainten
ance-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", 'se
t 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(sha
de) # 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
```

A

```
Tbs<-onelump_varenv(t=120,time=times3[2],Tc_init=Tc_init,thresh = 30, AMASS = m
ass, 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(a
ctfeed)

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_p
res","cumrepro","cumbatch","p_B_past","O2FLUX","CO2FLUX","MLO2","GH2OMET
","DEBQMET","DRYFOOD","FAECES","NWASTE","wetgonad","wetstorage","wetfo
od","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, d
el_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=debou
t[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 fr
ame (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 ... result
sN'
  namh <- paste("turtles", sc, sep = "")  #generate new name with added sc count
  rassh<-assign(namh,turtles) #assign new name to results. call 'results1, results2 ... resu
ltsN'
  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 activit
y 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,".c
sv\"",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
```

A

```
    NLCommand(paste("export-plot \"Movement costs\" \"",results.path,lfh,".csv\"",se
p=""))
  }
} # ********************** 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"
```

A

Figures

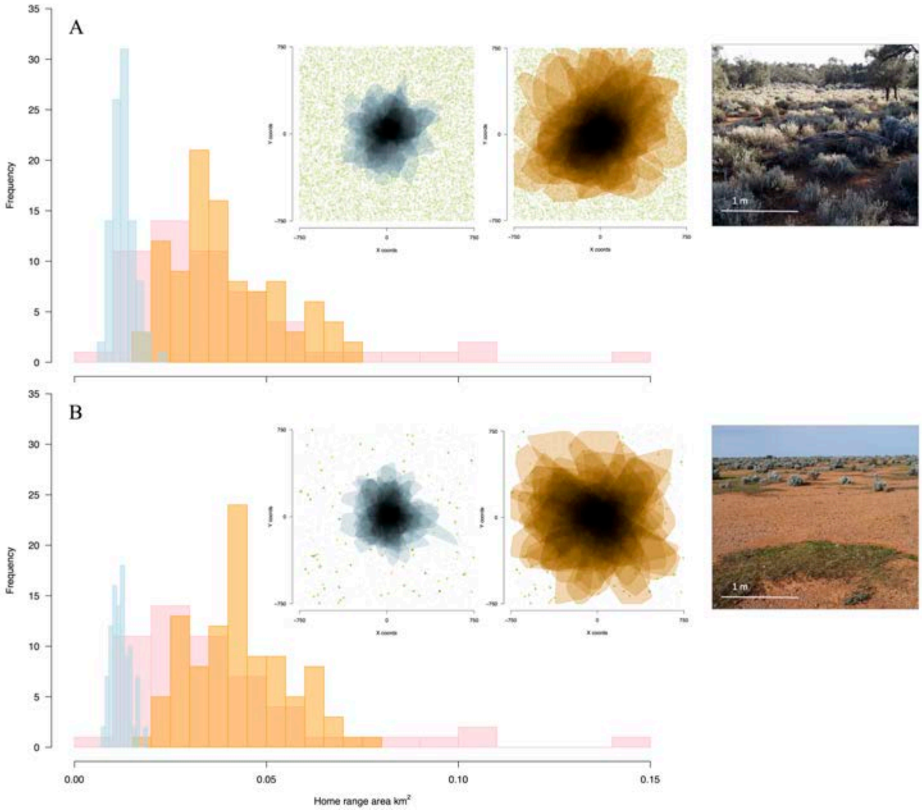

Figure A1. Distributions of home range area (km^2) of real animals (pink) and simulated optimising (orange) and satisficing (blue) movement strategies under (A) dense and (B) sparse resource distribution (food and shade). Insets (L–R): Home range polygons in space showing overlap of simulated satisficing (blue) and optimising (orange) movement strategies, and examples of (upper) dense and (lower) sparse resource distributions in the study site.

A

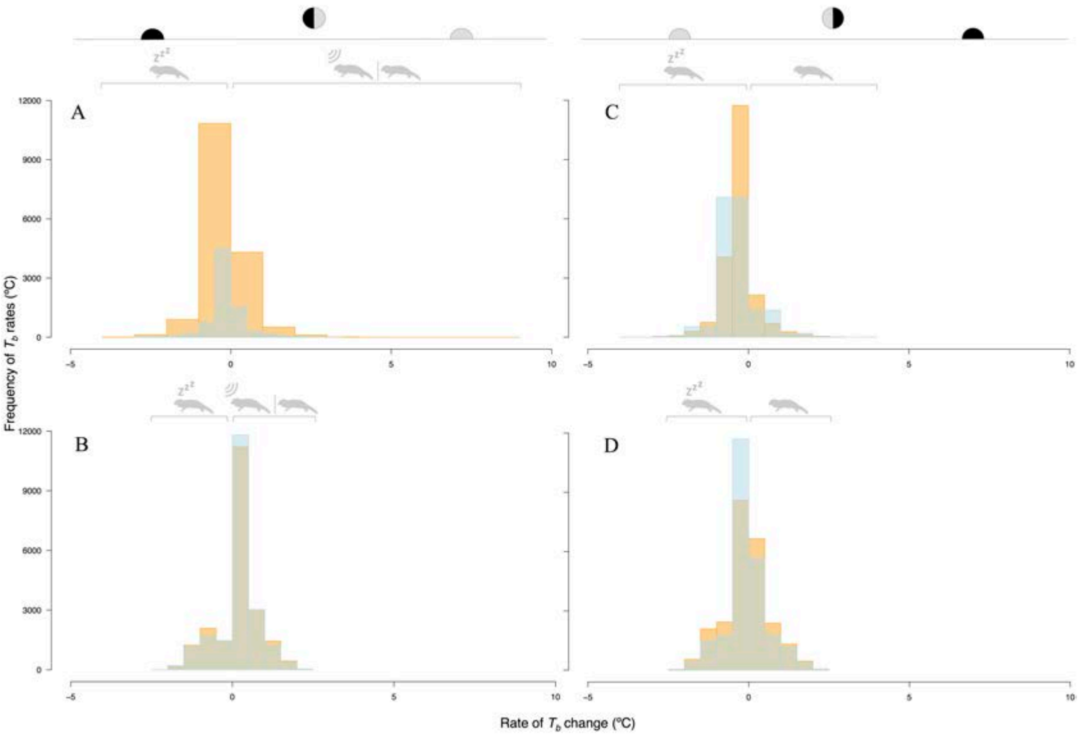

Figure A2. Rates of T_b change (ºC 2 min^{-1}) comparing (A) observed active (orange; #11885) and passive (blue; #11533) movement and (B) simulated optimising (orange) and satisficing (blue) movement for the morning hours (heating period; 06:00–12:00). (C) Observed active (orange) and passive (blue) 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. 5).

References

Auburn, Z. M., Bull, C. M. and Kerr G. D. 2009. The visual perceptual range of a lizard, *Tiliqua rugosa*, *Journal of Ethology*, pp. 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.

A