Supporting Information

for

Thermal performance curves and the Metabolic Theory of Ecology – a practical guide to models and experiments for parasitologists

Péter K. Molnár, Jason P. Sckrabulis, Karie A. Altman, Thomas R. Raffel

CONTENT:

Appendix S1. Anderson-May-type model for the host-parasite dynamics of a one-nematode-one-endotherm-host system (Fig. 1B).

In this Appendix, we specify a dynamic host-parasite model for the one-nematode-one-endotherm-host dynamics shown in Figure 1B. Here, adult parasites reside within the endotherm host and produce transmission stages that pass out of the host into the environment. Free-living larvae then need to pass through one or more developmental stages before they become infective and can be taken up again by the host. All transition rates may hereby depend on temperature if they occur outside the definitive host, but are assumed temperature-independent otherwise due to host endothermy (Fig. 1B). Following the classical framework of Anderson & May, we obtain a set of coupled differential equations that explicitly track the abundance of hosts, (H), adult parasites within hosts (P), and free-living uninfective (L_U) and infective (L_I) larvae as a function of temperature (T) through time (t) (Anderson & May, 1978; Molnár et al., 2013);

(S1a) $$\frac{dL_U}{dt} = \lambda_P P - \mu_U(T)L_U - \theta_U(T)L_U$$

(S1b) $$\frac{dL_I}{dt} = \theta_U(T)L_U - \mu_I(T)L_I - \rho_H(T)HL_I,$$

(S1c) $$\frac{dP}{dt} = q_H \rho_H(T)HL_I - (\mu_P + d_H)P - \delta_P H\left(\frac{P}{H} + \frac{P^2}{H^2}\frac{j+1}{j}\right)$$

(S1d) $$\frac{dH}{dt} = (b_H - d_H)H - (\beta_P + \delta_P)P$$

Here, λ_P represents the *per capita* rate of parasite birth; $\mu_U(T)$ and $\mu_I(T)$ are the stage- and temperature-dependent *per capita* rates of parasite mortality; $\theta_U(T)$ is the temperature-dependent *per capita* development rate of uninfective parasite larvae to the infective stage; $\rho_H(T)H$ is the rate at which an infective larva encounters hosts (temperature-dependent for parasites that actively seek out host, and temperature-independent otherwise); q_H is the probability that a parasite establishes within its host following uptake; b_H and d_H are the rates of host birth and natural host death; β_P and δ_P quantify the effect of the parasite on host fecundity and death; and j describes the degree of parasite aggregation within the host population assuming a negative binomial distribution. Seasonal variability and potential range changes are ignored in this model for simplicity, but the model could easily be extended with these and other processes for more detailed analyses (Anderson & May, 1991; Keeling & Rohani, 2008).

$R_0(T)$, defined as the temperature-dependent expected lifetime reproductive output of a newborn larva in the absence of density-dependence (Anderson & May 1991), is then calculated as the dominant eigenvalue of the temperature-dependent next-generation matrix, $N(T)$, of equations S1 (Hurford et al., 2010). Here, $N(T) = F(T)V(T)^{-1}$ with

$$F(T)=\begin{pmatrix}0 & 0 & \lambda_P\\ 0 & 0 & 0\\ 0 & 0 & 0\end{pmatrix} \quad \text{and} \quad V(T)=\begin{pmatrix}\mu_U(T)+\theta_U(T) & 0 & 0\\ -\theta_U(T) & \mu_I(T)+\rho_H(T)H & 0\\ 0 & -q_H\rho_H(T)H & \mu_P+d_H+\delta_P\end{pmatrix},$$

from which we obtain

$$R_0(T)=\frac{q_H\lambda_P}{\mu_P+d_H+\delta_P}\cdot\frac{\theta_U(T)}{\mu_U(T)+\theta_U(T)}\cdot\frac{\rho_H(T)H}{\mu_I(T)+\rho_H(T)H} \quad .$$

Appendix S2. Why apparent development time is more temperature-sensitive than real development time

In this Appendix, we consider the development of uninfective free-living larvae to infectivity (Fig. 1B; stage L_U to L_I), and show that estimates of the activation energy of observed (apparent) development times, E_θ', will systematically overestimate the true activation energy of development, E_θ, due to a failure to account for the influence of mortality.

First, we note that the observed (apparent) distribution of development times will underestimate the actual distribution of development times in any cohort experiment if mortality is not accounted for (Braner & Hairston, 1989). More specifically, it can be shown that the difference between the true and apparent means of development time, M and M', equals the product of the variance in development time, σ^2, and the mortality rate, μ, if development times are normally distributed (Braner & Hairston, 1989) (Fig. S1A):

(S2) $$M - M' = \sigma^2 \mu \ .$$

As this difference increases with increasing temperatures (due to the temperature-dependence of the mortality rate, $\mu(T)$), estimates of E_θ' (apparent activation energy of development, based on $M'(T)$) will always overestimate the true activation energy of development E_θ (which needs to be estimated from the true development means, $M(T)$; Fig. S1B,C).

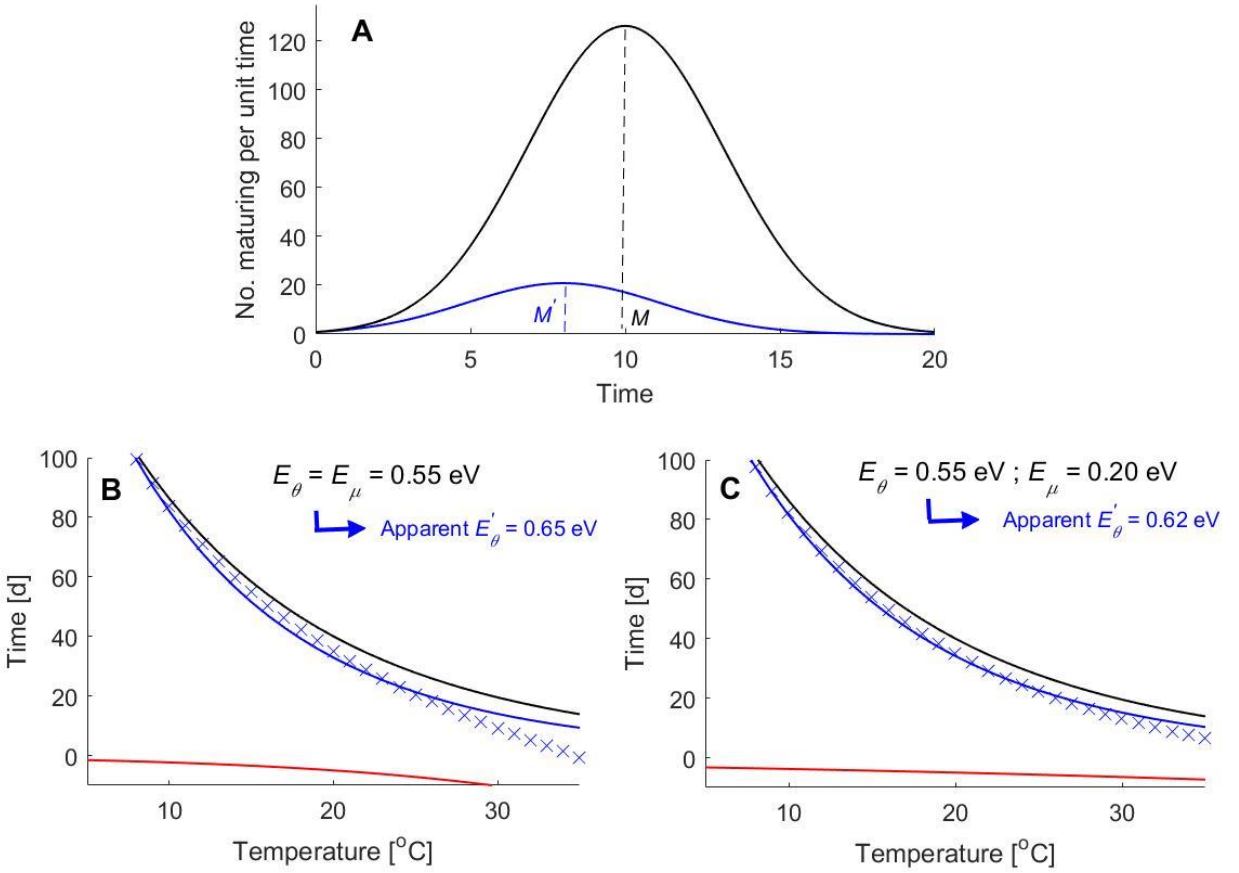

Figure S1. Why apparent development time is more temperature-sensitive than real development time. (A) The relationship between apparent and real development. Number of individuals maturing as a function of time when development time is normally distributed, without (=real development; black line) and with mortality (=apparent development; blue line). Vertical lines mark the mean development time without (M) and with ($M' = M - \sigma^2\mu$) mortality. Parameters were set as M = 10 d, σ^2 = 10 d^2 (variance of development time), and μ = 0.2 d^{-1} (mortality rate), implying M' = 8 d (eqn S2). Redrawn from Braner & Hairston (1989). Panels (B) and (C) illustrate why these discrepancies lead to biased (overestimated) activation energies of development time if these estimates are based on apparent development time. In both panels we assume that development time is normally distributed with a mean $M(T)$ that is temperature-sensitive according to the BA equation with activation energy E_θ = 0.55 eV, variance σ^2 = 10 d^2, and a baseline development time of $1/\theta_0$ = 40 d at T_0 = 20 C (black line). Furthermore, we assume that mortality rates also follow a BA equation with E_μ = 0.55 eV (panel B) or E_μ = 0.20 eV (panel C), and a baseline mortality of μ = 0.5 d^{-1} at T_0 = 20 C. Adding $-\sigma^2\mu(T)$ (red lines) to $M(T)$ yields apparent development times as a function of temperature, $M'(T)$ (eqn 5; blue crosses). Fitting a BA curve (eqn 1) to these simulated data (blue lines) yields activation energies E_θ' that overestimate the true activation energy of development E_θ, essentially because the contributions of mortality and developmental variance (red lines) lead to steeper temperature-dependencies in apparent development times (blue crosses) than in true development times (black line). This effect becomes stronger as mortality becomes more temperature-sensitive (contrast panels B & C; c.f. Fig. 5).

Appendix S3. The 'classical approach to perturbation analyses of $R_0(T)$ (Fig. S2), and R-codes for both the classical approach and the new approach suggested in Section 3.1

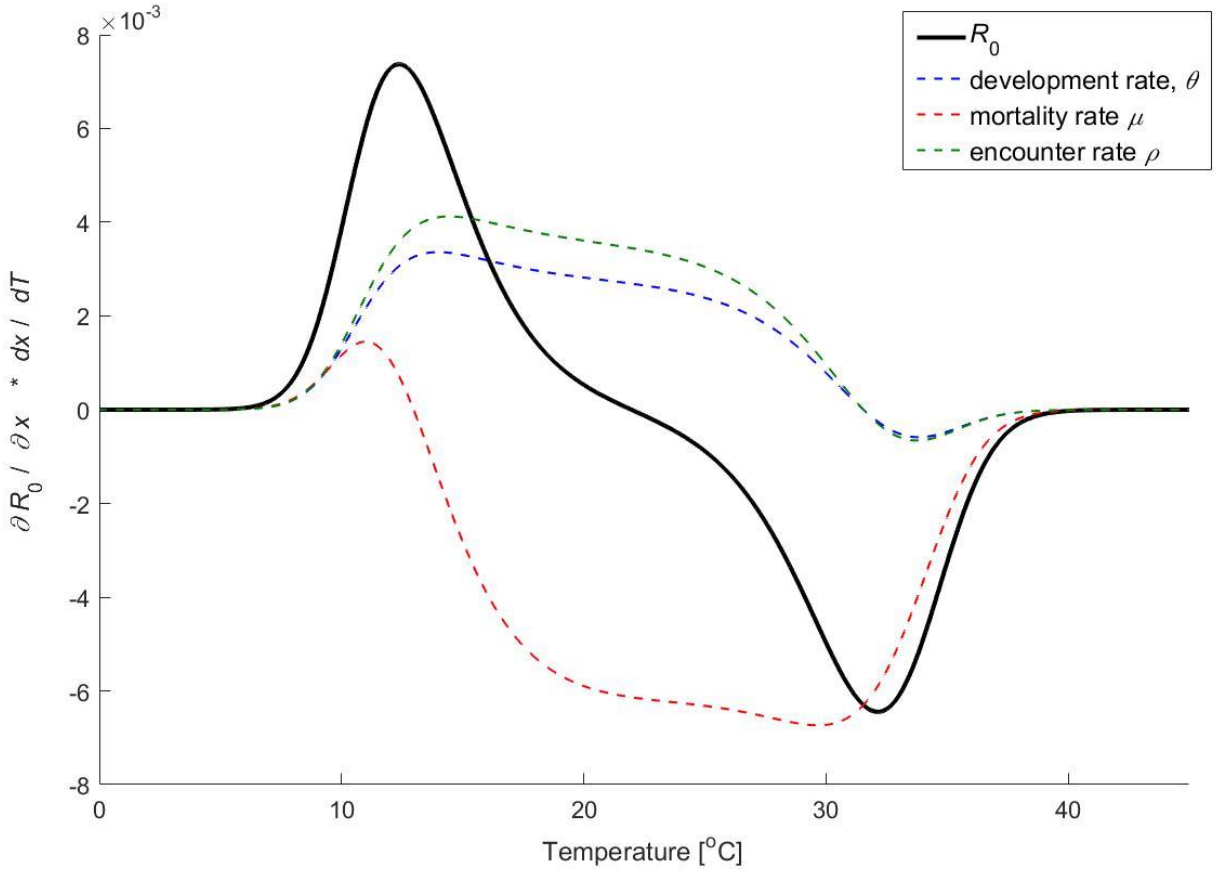

Figure S2. Perturbation analyses of an MTE-parameterized null model for a nematode with an endotherm host, a mandatory free-living stage, and active host infection (cf. Figs. 1, 3), but now using the 'classical approach' of determining the additive contributions of development (θ), mortality (μ), and host encounter (ρ) to the temperature sensitivity of $R_0(T)$, dR_0/dT. The additive contributions of each of these parameters are hereby calculated using the chain rule of partial derivatives, i.e. $dR_0/dT = \partial R_0/\partial\theta * d\theta/dT + \partial R_0/\partial\mu * d\mu/dT + \partial R_0/\partial\rho * d\rho/dT$. Black curve: sensitivity of $R_0(T)$ to temperature, dR_0/dT; blue (development), red (mortality), and green (encounter) curves show the additive contribution of each trait to this temperature-sensitivity (i.e. $\partial R_0/\partial x * dx/dT$). Baseline parameters of development (θ), mortality (μ) and encounter (ρ) rates are set as in Fig. 3: $\theta_0 = 0.03$ d^{-1}, $\mu_0 = 0.06$ d^{-1}, $\rho_0 = 0.01$ d^{-1} at $T_0 = 22.5$ C, and $E = 0.65$ eV, $-E^L = E^H = 3.25$ eV, $T^L = 10$ C, $T^H = 35$ C for all three traits.

S3.1 – R-code for the 'classical approach' to perturbation analyses of $R_0(T)$ (Fig. S2)

```
# The 'classical approach' to elasticity analyses of R0(T). This code
# reproduces Fig S2 of the article, that is, it decomposes the
# derivative of R0 w.r.t. temperature T, into its partial derivative
# w.r.t to the performance traits development, mortality, encounter
# rate, using the chain rule of partial derivations.

# Parameter settings ------------------------------------------------

minTemp <- 273.15   # setting min and max temperatures for simulations
maxTemp <- 318.15
resol <- 0.01
TT <-  seq(minTemp, maxTemp, resol)

theta0 <- 0.03; mu0 <- 0.06; rho0 <- 0.01      # baseline parameters
E_theta <- 0.65; E_mu <- 0.65; E_rho <- 0.65
T0_theta <- 295.65; T0_mu <- 295.65; T0_rho  <- 295.65
T0_thetaL <- 283.15; T0_muL <- 283.15; T0_rhoL <- 283.15
T0_thetaH <- 308.15; T0_muH <- 308.15; T0_rhoH <- 308.15
E_thetaL  <- -3.25; E_muL <- -3.25; E_rhoL <- -3.25;
E_thetaH <- 3.25; E_muH <- 3.25; E_rhoH  <- 3.25;
k <- 8.62*10^(-5)                              # Boltzmann's constant


# Sharpe-Schoolfield expressions for theta, mu, rho and Ro ----------

thetaSS <- quote(theta0 * exp(-(E_theta/k) * (1/T_ - 1/T0_theta)) *
(1+ exp(E_thetaL/k * (-1/T_+1/T0_thetaL)) + exp(E_thetaH/k *
(-1/T_+1/T0_thetaH)))^(-1))
muSS <-  quote(mu0    * exp(-(E_mu   /k) * (1/T_ - 1/T0_mu))    * (1+
exp(E_muL /k   * (-1/T_+1/T0_muL))    + exp(E_muH/k    *
(-1/T_+1/T0_muH))))
rhoSS <-  quote(rho0   * exp(-(E_rho  /k) * (1/T_ - 1/T0_rho))   * (1+
exp(E_rhoL/k   * (-1/T_+1/T0_rhoL))   + exp(E_rhoH/k   *
(-1/T_+1/T0_rhoH)))^(-1))

R0SS <- quote((rhoSS*thetaSS) / ((muSS + thetaSS) * (muSS + rhoSS)))

# Substitute the SS-models for theta, mu, rho into R0
R0SS <- substituteDirect(R0SS, list(thetaSS = thetaSS, muSS = muSS,
rhoSS = rhoSS))


# Derivatives of R0 and SS parameters with respect to temperature ----

dthetaSS_dT <- deriv(thetaSS, "T_")
dmuSS_dT <- deriv(muSS, "T_")
drhoSS_dT <- deriv(rhoSS, "T_")
dR0SS_dT <- deriv(R0SS, "T_")
```

```
# Create functions that evaluate the derivatives at given temperatures

f_dthetaSS_dT <- function(T) attr(eval(dthetaSS_dT, list(T_=T)),
"gradient")
f_dmuSS_dT <- function(T) attr(eval(dmuSS_dT, list(T_=T)), "gradient")
f_drhoSS_dT <- function(T) attr(eval(drhoSS_dT, list(T_=T)),
"gradient")
f_dR0SS_dT <- function(T) attr(eval(dR0SS_dT, list(T_=T)), "gradient")

# Derivatives of R0 with respect to the performance traits theta, mu,
# rho ------------------------

R0 <- quote((rho*theta)/((mu+theta)*(mu+rho)))

dR0_dtheta <- deriv(R0, "theta")
dR0_dmu <- deriv(R0, "mu")
dR0_drho <- deriv(R0, "rho")

# Substitute the SS-models for theta, mu and rho into the derivatives

dR0_dtheta <- substituteDirect(dR0_dtheta[[1]], list(theta = thetaSS,
mu = muSS, rho = rhoSS))
dR0_dmu <- substituteDirect(dR0_dmu[[1]], list(theta = thetaSS, mu =
muSS, rho = rhoSS))
dR0_drho <- substituteDirect(dR0_drho[[1]], list(theta = thetaSS, mu =
muSS, rho = rhoSS))

# Create functions that evaluate the derivatives at given temperatures

f_dR0_dtheta <- function(T) attr(eval(dR0_dtheta, list(T_=T)),
"gradient")
f_dR0_dmu <- function(T) attr(eval(dR0_dmu, list(T_=T)), "gradient")
f_dR0_drho <- function(T) attr(eval(dR0_drho, list(T_=T)), "gradient")

# Calculate the contributions of theta (development), mu (mortality),
# and rho (encounter rate) to dR0/dT

diffR0_dT <- f_dR0SS_dT(TT)
Cont_theta <- f_dR0_dtheta(TT)*f_dthetaSS_dT(TT)
Cont_mu <- f_dR0_dmu(TT)*f_dmuSS_dT(TT)
Cont_rho <- f_dR0_drho(TT)*f_drhoSS_dT(TT)
```

```
# Plotting the figures ------------------------------------------------
xticks <- seq(minTemp, maxTemp, 10)
xticklabels <- xticks-273.15
legendlabels <- c(expression(italic(R[0])),
                  expression(paste("development rate ", theta, sep =
""")),
                  expression(paste("mortality rate ", mu, sep = "")),
                  expression(paste("encounter rate ", rho, sep = "")))
cols <- c("black", "blue", "red", "green")

curve(f_dR0SS_dT(x), minTemp, maxTemp, xlab = "", ylab = "", xaxt =
"n", col = cols[1])
curve(f_dR0_dtheta(x)*f_dthetaSS_dT(x), minTemp, maxTemp, lty=2,
col=cols[2], add=T)
curve(f_dR0_dmu(x)*f_dmuSS_dT(x), minTemp, maxTemp, lty=2,
col=cols[3], add=T)
curve(f_dR0_drho(x)*f_drhoSS_dT(x), minTemp, maxTemp, lty=2,
col=cols[4], add=T)

axis(1, at = xticks, labels = xticklabels)
title(xlab = "Temperature (˚C)",
      ylab =
expression(partialdiff*italic(R[0])/partialdiff*italic(x)%*%italic(d*x
)/italic(d*T)))
legend("topright", legend = legendlabels, col = cols, lty =
c(1,2,2,2), cex = 0.75)
```

S3.2 – R-code for the perturbation analyses approach of $R_0(T)$ suggested in Section 3.1 (Fig. 6)

```
# Elasticity analyses of R0(T) with respect to the two metrics (AUC,
# skewness) in section 3.1. This code reproduces Figure 6 of the
# article, that is, it calculates the relative change in the area
# under the curve and skewness of R_0(T) for a proportional change in
# the activation energies, inactivation energies, and baseline
# performance values of the Sharpe-Schoolfield models for the
# performance traits that underlie R0(T).

# Parameter settings ------------------------------------------------

minTemp <- 273.15   # setting min and max temperatures for simulations
maxTemp <- 318.15
resol <- 0.01
TT <-  seq(minTemp, maxTemp, resol)         # temperature vector for
                                            # calculations

theta0 <- 0.03; mu0 <- 0.06; rho0 <- 0.01      # baseline parameters
E_theta <- 0.65; E_mu <- 0.65; E_rho <- 0.65
T0_theta <- 295.65; T0_mu <- 295.65; T0_rho  <- 295.65
T0_thetaL <- 283.15; T0_muL <- 283.15; T0_rhoL <- 283.15
T0_thetaH <- 308.15; T0_muH <- 308.15; T0_rhoH <- 308.15
E_thetaL  <- -3.25; E_muL <- -3.25; E_rhoL <- -3.25;
E_thetaH <- 3.25; E_muH <- 3.25; E_rhoH  <- 3.25;
k <- 8.62*10^(-5)                              # Boltzmann's constant

# Create the elasticity matrices. The number of columns corresponds
# to the number of parameters, and the number of rows is the number
# of relative differences to be tested (here, 0%, -20%, +20%)

Elast_Skew <- matrix(0, nrow = 3, ncol = 12)
Elast_AUC <- matrix(0, nrow = 3, ncol = 12)

# Set a vector containing the original parameter values

Base_Params_Orig <- c(theta0, E_theta, E_thetaL, E_thetaH, mu0, E_mu,
E_muL, E_muH, rho0, E_rho, E_rhoL, E_rhoH)

# Define the vector of relative differences in parameter values that
# will be tested

Perc_change <- c(1, 0.8, 1.2)
```

```
# Definition of SS-models for development (theta), mortality (mu),
# encounter rate (rho), and the resulting R0(T) (cf. eqn 3; the
# temperature-independent constant C is ignored as it cancels out in
# the elasticity analyses)----------------------------

R0_fun <- function(Temp, Pars) {
  theta_SS <- Pars[1] * exp(-Pars[2]/k * (1/Temp-1/T0_theta)) *
(1+exp(Pars[3]/k * (-1/Temp+1/T0_thetaL)) + exp(Pars[4]/k *
(-1/Temp+1/T0_thetaH)))^(-1)
  mu_SS <- Pars[5] * exp(-(Pars[6]/k) * (1/Temp - 1/T0_mu)) *
(1+ exp(Pars[7]/k * (-1/Temp+1/T0_muL))   + exp(Pars[8]/k *
(-1/Temp+1/T0_muH)))
  rho_SS <-  Pars[9] * exp(-(Pars[10]/k) * (1/Temp - 1/T0_rho)) *
(1+ exp(Pars[11]/k * (-1/Temp+1/T0_rhoL)) +  exp(Pars[12]/k *
(-1/Temp+1/T0_rhoH)))^(-1)

  R0_SS <- theta_SS/(mu_SS + theta_SS)*rho_SS/(mu_SS+rho_SS)
  return(R0_SS)
}

# Calculating the relative changes in R_0 for each parameter variation

for (i in 1:length(Base_Params_Orig)) {
  for (j in 1:length(Perc_change)) {
    Base_Params <- Base_Params_Orig

    # Change parameter i in the parameters vector
    Base_Params[i] <- Base_Params[i]*Perc_change[j]

    # Calculate the new R_0 for all temperatures
    R0_SS <- R0_fun(TT, Base_Params)

    # Determine at which temperature the maximum R0 is reached
    max_index <- which.max(R0_SS)
    T_pk <- TT[max_index]

    # Calculate the value of the 2 objective functions (skewness, AUC)
    skew = (T_pk-T0_thetaL)/(T0_thetaH-T0_thetaL)
    AUC1 = integrate(R0_fun, minTemp, maxTemp, Pars =
      Base_Params)$value

    if (j == 1) {
      Elast_Skew[j, i] <- skew
      Elast_AUC[j, i] <- AUC1
    } else {
      Elast_Skew[j, i] <- skew/Elast_Skew[1, i]-1
      Elast_AUC[j, i] <- AUC1/Elast_AUC[1, i]-1
    }
  }
}
```

```
# Plotting the Figures ------------------------------------------------

# Create a matrix for each of the second and third row in the
# elasticity matrices. This is needed for the barplots.
EA1 <- t(array(Elast_AUC[2,],c(4,3)))
EA2 <- t(array(Elast_AUC[3,],c(4,3)))
ES1 <- t(array(Elast_Skew[2,],c(4,3)))
ES2 <- t(array(Elast_Skew[3,],c(4,3)))

AUC_yrange <- c(min(EA1, EA2), max(EA1, EA2))
Skew_yrange <- c(min(ES1, ES2), max(ES1, ES2))
Par_labels <- expression(italic(y[0]),
                         italic(E),
                         italic(E[L]),
                         italic(E[H]))
legendlabels <- expression(paste("development rate ", theta, sep =
""),
                           paste("mortality rate ", mu, sep = ""),
                           paste("encounter rate ", rho, sep = ""))
plotcolors <- topo.colors(3)

barplot(EA1, ylim = AUC_yrange , names.arg = Par_labels, beside = T,
col = plotcolors, ylab = "Relative change in AUC")
barplot(EA2, beside = T, add=T, col = plotcolors, density = 15)
legend("topright", legend = legendlabels, fill = plotcolors, cex =
0.75)

barplot(ES1, ylim = Skew_yrange, names.arg = Par_labels, beside = T,
col = plotcolors, ylab = "Relative change in Skewness")
barplot(ES2, beside = T, add=T, col = plotcolors, density = 15)
legend("topright", legend = legendlabels, fill = plotcolors, cex =
0.75)
```

Appendix S4. Case study in SS model fitting using *Schistosoma mansoni* cercaria activity data

Here, we provide a detailed case study showing how to fit the SS model to thermal performance data (here, cercaria swimming speed), following the maximum likelihood approach of Régnière et al. (2012) and implementing it in R (R Development Core Team, 2014).

S4.1 – Study species

The trematode parasite *Schistosoma mansoni* is the most widespread parasitic agent of human schistosomiasis (Colley et al., 2014). The parasite has a two-host life cycle, where free-swimming cercariae seek out humans after being released from the intermediate host snail *Biomphalaria glabrata* (Colley et al., 2014) (Fig. 1C, D). We conducted an experiment to quantify the temperature dependence of *S. mansoni* cercaria swimming speed as a proxy for parasite metabolism. The methods for data collection are provided in Section S4.2, and the data fitting procedures are detailed in the subsequent sections.

S4.2 – Experimental design and data collection

S. mansoni-infected *B. glabrata* snails were obtained from the Schistosomiasis Resource Center (BEI Resources, NIAID, NIH). Snails were maintained at room temperature (22 C) in artificial spring water (Cohen et al., 1980) with a 12:12 hour photoperiod, Snails were fed *ad libitum* a combination of frozen spinach and "snail jello", which is an agar preparation that includes calcium powder and ground Tetramin™ fish flakes (Paull et al., 2015). Prior to cercaria collection, five infected snails were moved to a temperature-controlled Styrofoam incubator (Raffel et al., 2013) and were maintained there at 22 C for one week. Two days prior to the experiment (day 6 of acclimation), a wooden panel was placed over the clear lid of the incubator to maintain complete darkness (0:24 photoperiod) for the remaining 2 days. Snails were then placed under a bright LED light for 1 hour to encourage cercaria release. *S. mansoni* cercariae were collected and fluorescently stained with a fluorescently-labeled fatty acid analogue (BODIPY FL C12; Invitrogen, Carlsbad, CA, USA) following the protocol of LaFonte et al. (2015). The experiment was conducted in two temporal blocks. All cercariae were observed within 3 hours of release from the snail (n=49). The order in which cercariae were exposed to temperature treatments was randomized within temporal blocks to account for potential changes in cercariae viability throughout the day.

Individual cercaria were then pipetted into the depression on the underside of a specimen cup with approximately 5 µL of 22 C artificial spring water (Fig. S3). The lid was then floated for five minutes in 250 mL of water, maintained at one of eight temperatures (13, 16, 19, 22, 25, 28, 31, 34 C) to ensure that the water in the depression containing the cercaria had reached and stayed at the target temperature. We verified that this timing was sufficient using an infrared thermometer prior to the experiment. Each cercaria was viewed under a fluorescent stereomicroscope (Leica MZ10F, Leica Microsystems, Wetzler, Germany) fitted with a color digital video camera (Leica DFC450C, Leica Microsystems, Wetzler, Germany). Video recordings of cercaria activity were obtained using the MicroManager software (Edelstein et al., 2014) interacting with VirtualDub screen capture software (www.virtualdub.org). Cercaria swimming speed was quantified by tracking the position of each cercaria's tail base throughout each video and calculating the velocity between successive points in millimeters per second ($mm \cdot sec^{-1}$).

S4.3 – Loading and plotting data

The following code loads the experimental data (performance temperature, cercaria swimming speeds, temporal block) into the R console and creates a dataframe named "myData".

```
#Experimental performance temperatures (°C)
PerfTemp <- c(22, 22, 19, 19, 22, 19, 25, 13, 13, 25, 25, 16, 31, 16, 16, 31,
     31, 34, 34, 28, 28, 28, 34, 34, 34, 28, 34, 28, 34, 34, 28, 31, 16, 31,
     16, 16, 31, 19, 19, 22, 22, 19, 25, 13, 13, 25, 13, 13, 25)

#Cercaria swimming speeds (mm/sec)
Velocity <- c(0.833, 0.209, 0.201, 0.115, 0.183, 0.143, 0.611, 0.165,
     0.199, 0.597, 0.416, 0.097, 0.208, 0.292, 0.159, 0.370, 0.228, 0.152,
     0.178, 0.502, 0.516, 0.480, 0.270, 0.191, 0.255, 0.508, 0.248, 0.585,
     0.215, 0.115, 0.601, 0.165, 0.181, 0.268, 0.289, 0.122, 0.241, 0.173,
     0.194, 0.180, 0.369, 0.406, 0.281, 0.198, 0.302, 0.449, 0.208, 0.052,
     0.417)

#Temporal blocks in the experiment
#Random effects must be entered as numeric vectors for this model formulation
BlockReal <- c(1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
     1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
     2, 2, 2, 2)

myData <- as.data.frame(cbind(PerfTemp,Velocity))
myData$PerfTemp <- as.double(PerfTemp)       #numeric variable
myData$Velocity <- as.double(Velocity)       #numeric variable
```

These data can now be plotted using

```
plot(PerfTemp,Velocity)
```

Note that the data clearly extend beyond T_{pk}, capturing the high-temperature inactivation (Fig. 7B). We therefore proceed with fitting the SS model to these thermal performance data, and not the simpler BA-model, which is unable of capturing these nonlinearities.

S4.4 – Fitting the Sharpe-Schoolfield (SS) model to thermal performance data using maximum likelihood

Our dataset did not include temperatures close enough to the critical thermal minimum of *S. mansoni*, so we could not estimate the low-temperature inactivation energy (E^L) or the corresponding threshold (T^L). We therefore simplified the SS model by omitting these terms (i.e. setting T^L=0 K) and describe the temperature-dependence of cercaria performance $p(T)$ within the experimental temperature range using

$$\text{(S3)} \qquad p(T) = p_0 e^{-\frac{E}{k}\left(\frac{1}{T}-\frac{1}{T_0}\right)} \cdot \left[1 + e^{\frac{E^H}{k}\left(\frac{1}{T^H}-\frac{1}{T}\right)}\right]^{-1}.$$

The following code implements the maximum likelihood approach of Régnière et al. (2012) in R, estimating the activation energy (E), high-temperature inactivation energy (E^H) and temperature threshold (T^H) of cercaria swimming speed.

```
#Define constants
k  <- 8.62*10^(-5)      #Boltzmann's constant
K  <- 273.15            #Celsius to Kelvin conversion
To <- 19 + K            #Standardization temperature

#Log-likelihood function to be optimized, assuming a log-normal error
#distribution (after Régnière et al., 2012)

SSmodel <- function(data, par){
   s_epsilon <- par[1]
   pTo <- par[2]
   E   <- par[3]
   Th  <- par[4]
   Eh  <- par[5]

   se2 <- (-0.5)*s_epsilon^2  #mean of the normally distributed variable
                              #'epsilon' below (Régnière et al., 2012)
   x <- data[,1]+K
   y <- data[,2]

   expected <- pTo*exp(-(E/k*(1/x-1/To)))*(1+exp(Eh/k*(1/Th-1/x)))^-1
   epsilon <- log(y/expected)       #note that errors are calculated as
                                    #ratios, not differences

   p <- dnorm(epsilon,se2,s_epsilon)

   for (i in 1:length(p)){
      if(p[i] < 1e-10) p[i] <- 1e-10     #prevents log(0) computation errors
   }

   LL<- sum(log(p))
   -LL
}

#'Reasonable' initial parameter estimates to ensure convergence of the
#optimization procedure
init <- c(s_epsilon= 0.1, pTo= 0.2, E= 0.65, Th= 303, Eh= 3.5)


#Set parameter lower and upper parameter boundaries
lower <- c(0, 0, 0, 0, 0)
upper <- c(Inf, Inf, Inf, Inf, Inf)

#Optimize full model for all parameters
model <- optim(par= init, fn= SSmodel, data= myData,
   method= "L-BFGS-B", lower= lower, upper= upper, hessian=TRUE)
model                                    #view model output

#Output optimized model coefficients and their 95% confidence intervals
Coef <- model$par
SE <- sqrt(diag(solve(model$hessian)))    #SE for parameter estimates
Value <- SE*1.96               #95% confidence intervals for estimates
```

```
cbind(Coef, Value)

#Table of model parameter estimates and the value to calculate 95% confidence
#intervals (Coef +/- Value)
#                 Coef      Value
#s_epsilon   0.4197777 0.08310917
#pTo         0.2703667 0.04715787
#E           0.6984673 0.29862724
#Th        303.4948870 2.91925910
#Eh          3.2046242 1.32339403

#Plot the optimized model function over data (Fig. 7B, solid black line).
plot(PerfTemp, Velocity)             #plot data
cercSwim <- function(x, pTo, E, Eh, Th){
   y = pTo*exp(-(E/k*(1/x-1/To)))*(1+exp(Eh/k*(1/Th-1/x)))^-1
}
LineTemps <- seq(13, 34, length= 100) + K
PredictionsDnorm <- cercSwim(x= LineTemps, pTo= 0.270 , E= 0.698 , Eh= 3.205
, Th= 303.5)
lines(LineTemps-K, PredictionsDnorm)   #plot optimized SS model
```

The optimized parameter values of the SS-model and their confidence intervals are provided in the table above, and the model fit to data is shown in Fig. 7B. Our parameter estimates conformed closely to *a priori* expectations based on the interspecific means of activation energies (E = 0.65 eV; Gillooly et al., 2001; Brown et al., 2004) and inactivation energies (E^H = 5 × 0.65 eV = 3.25 eV; Molnár et al., 2013), although we note that this need not be the case in general (Section 2). However, our estimate of the high-temperature inactivation threshold T^H = 303.49 ± 2.92 K (= 30.34 ± 2.92 C) for cercaria swimming speed was nearly 15 C lower than the literature estimate of CT_{max} = 45 C for *S. mansoni* (Lawson & Wilson, 1980). This is probably because our measure of parasite performance differed from that assessed by Lawson and Wilson (1980), who measured the temperature needed to induce 100% cercaria mortality. Schistosome cercariae use temperature as one way of detecting definitive hosts, and they might interpret environmental temperatures similar to that of their definitive host's body temperature (37 C) as an indication that a definitive host is nearby (Colley et al., 2014). For this reason, temperatures greater than 37 C might induce a behavioral response in which cercariae stop swimming and shed their tails in preparation to penetrate a definitive host (Colley et al., 2014). Indeed, during this experiment we frequently observed *S. mansoni* cercariae that stopped swimming and began crawling along the cup surface in the high temperature treatments (31 and 34 C), supporting this interpretation.

S4.5 – Adding a random effect to the SS model with log-normal error structure

In this final section, we modify the R code from Section S4.4 to allow for the addition of potential random effects in the sampling design. Such random effects may, for example, arise when experiments are conducted on individuals originating from separate source populations, when individuals have differing thermal histories, or when experiments are conducted in multiple temporal blocks or incubators. Régnière *et al.* (2012) showed how to incorporate a random effect into the likelihood function of the SS model, and provide example SAS code. The primary change to the previous code is the addition of a new parameter, `s_upsilon`, which represents the

standard deviation of the population means. For example, in an experiment with multiple blocks, this parameter would describe the standard deviation of the block means.

Our experiment with *S. mansoni* cercariae was conducted in 2 temporal blocks (cf. variable `BlockReal` in S4.3). However, model optimization showed that the random effect of block was negligible (`s_upsilon` < 0.0001) in our data[1]. A block effect so small is difficult to evaluate statistically and can make the model unstable, because the likelihood function approaches infinity as `s_upsilon` approaches zero. We therefore created an artificial blocking variable `BlockFake`) for demonstration purposes, where each data point was assigned to `BlockFake` = 1 if swimming speed was lower than the mean swimming speed at that performance temperature, and to `BlockFake` = 2 if it was higher (Figure S4).

```
#Artificial (fake) blocking variable for demonstration purposes

BlockFake <- c(2, 1, 1, 1, 1, 1, 2, 1, 2, 2, 1, 1, 1, 2, 1, 2, 1, 1, 1, 1, 1,
1, 2, 1, 2, 1, 2, 2, 2, 1, 2, 1, 1, 2, 2, 1, 1, 1, 1, 1, 2, 2, 1, 2, 2, 1, 2,
1, 1)

myData2 <- as.data.frame(cbind(PerfTemp,Velocity,BlockFake))
myData2$PerfTemp <- as.double(PerfTemp)         #numeric
myData2$Velocity <- as.double(Velocity)         #numeric

#Modified maximum likelihood approach, including a random effect:

#define constants
k <- 8.62*10^(-5)     #Boltzmann's constant (eV/K)
K <- 273.15           #Celsius to Kelvin conversion
To <- 19 + K          #Standardization temperature (arbitrary)

SSmodelrandom <- function(data, par){
  s_epsilon <- par[1]
  s_upsilon <- par[2]
  pTo <- par[3]
  E  <- par[4]
  Th <- par[5]
  Eh <- par[6]
```

[1] Optimizing this model with the real temporal block variable (BlockReal) required precise selection of initial parameter estimates and a more forgiving optimization method to achieve model convergence, due to the negligibly small magnitude of the random effect (s_upsilon ≈ 2.0×10^{-9}). The traditional "optim" function only has one available method for bounded parameter optimization ("L-BFGS-B"). To estimate s_upsilon for the BlockReal effect, we used the updated "optimx" function (package optimx) which allows bounded parameter estimation using the robust "spg" method (Nash & Varadhan, 2011; Nash, 2014). Such a small s_upsilon is problematic because the likelihood function approaches infinity as s_upsilon approaches zero (Régnière et al., 2012), leading to inflated likelihood values and thus potentially inaccurate estimates for other parameters. It is often easier to achieve model convergence with a simpler model (in this case, the earlier model with no random effect), and this can be a good way to refine starting values for more complex models that might have more trouble converging. In this case, plotting the data also helped to reveal the negligible difference between blocks, leading us to try out the smaller starting values for s_upsilon that ultimately led to model convergence.

```
  se2 <- (-0.5)*s_epsilon^2

  x <- data[,1]+K; y <- data[,2]; Block <- data[,3]

  expected <- pTo*exp(-(E/k*(1/x-1/To)))*(1+exp(Eh/k*(1/Th-1/x)))^-1
  data <- cbind(data, expected)
  upsilon <- numeric(length(x))

  for (j in 1:max(Block)){
    #upsilon_j is the mean deviation from expected for treatment j
    upsilon_j <- mean(subset(data, Block==j)[,2]/subset(data,
                 Block==j)[,4])
    #insert each datapoint's upsilon value into the dataframe
      for (i in 1:length(x)){
        if (Block[i]==j) upsilon[i] <- upsilon_j
      }
  }
  epsilon <- log(y/(expected*upsilon))      #See Régnière et al. 2012

  if(s_upsilon < 1e-20) s_upsilon <- 1e-20  #helps prevent NaN's
  if(s_epsilon < 1e-5) s_epsilon <- 1e-5

  p1 <- dnorm(upsilon, mean= 1, sd= s_upsilon)
  p2 <- dnorm(epsilon, mean= se2, sd= s_epsilon)

  for (i in 1:length(p1)){                  #helps prevent NaN's
    if(p1[i] < 1e-5) p1[i] <- 1e-5
    if(p2[i] < 1e-5) p2[i] <- 1e-5
  }

  LL <- sum(log(p1*p2))
  return(-LL)
}

#Initial parameter estimates and boundaries for parameter search space
init <- c(s_epsilon= 0.4, s_upsilon= 0.5, pTo= 0.2704665,
            E= 0.697644, Th= 303, Eh= 3.2059197)

lower <- c(0, 0, 0, 0, 0, 0)
upper <- c(Inf, Inf, Inf, Inf, Inf, Inf)

#Optimize full model for all parameters
model <- optim(par= init, fn= SSmodelrandom, data= myData2,
         method= "L-BFGS-B", lower= lower, upper= upper, hessian=TRUE)
model

#Output optimized model coefficients and their standard errors
coef <- model$par
SE <- sqrt(diag(solve(model$hessian)))   #SE for parameter estimates
Value <- SE*1.96                         #95% confidence intervals
cbind(coef,Value)
```

```
#Table of model parameter estimates and the value to calculate 95% confidence
#intervals (Coef +/- Value)
#                 coef       Value
# s_epsilon   0.2923503 0.05986718
# s_upsilon   0.2872339 0.05993339
# pTo         0.3010968 0.03805723
# E           0.6950091 0.22658316
# Th        303.2161098 2.26526098
# Eh          3.0731122 0.91245373


#Plot model
#subset the blocks
blockL <- subset(myData2,BlockFake=="1")
blockH <- subset(myData2,BlockFake=="2")
plot(blockH$PerfTemp,blockH$Velocity,
  pch=1,col='red',ylim=c(0,1), , bty="l",
  xlab="Temperature (C)", ylab="Cercaria swimming speed (mm/sec)")
points(blockL$PerfTemp,blockL$Velocity,pch=16,col='blue')

LineTemps<-seq(13,35,length=100)+K
Predictions<-cercSwim(x=LineTemps, pTo= 0.301 , E= 0.695 , Eh= 3.073 , Th=
303.216)          #full model predictions
PredictionsHigh<-Predictions+(Predictions*.287)       #high block predictions
PredictionsLow<-Predictions-(Predictions*.287)        #low block predictions
lines(LineTemps-K, PredictionsHigh, lwd=2, lty=3,col='red')
lines(LineTemps-K, PredictionsLow,lwd=2, lty=2,col='blue')
```

Including this artificial (fake) random effect, maximum likelihood estimated `s_epsilon` = 0.292 ± 0.060 mm·sec^{-1} and `s_upsilon` = 0.287 ± 0.060 mm·sec^{-1}, showing that the block effect accounted for about half of the random error in this dataset as expected. The remaining parameter estimates were similar to those obtained without a random effect (p_0 = 0.301 ± 0.038 mm·sec^{-1}; E = 0.695 ± 0.227 eV; T^H = 303.22 ± 2.27 K [= 30.07 ± 2.27 C]; E^H = 3.073 ± 0. 912 eV).

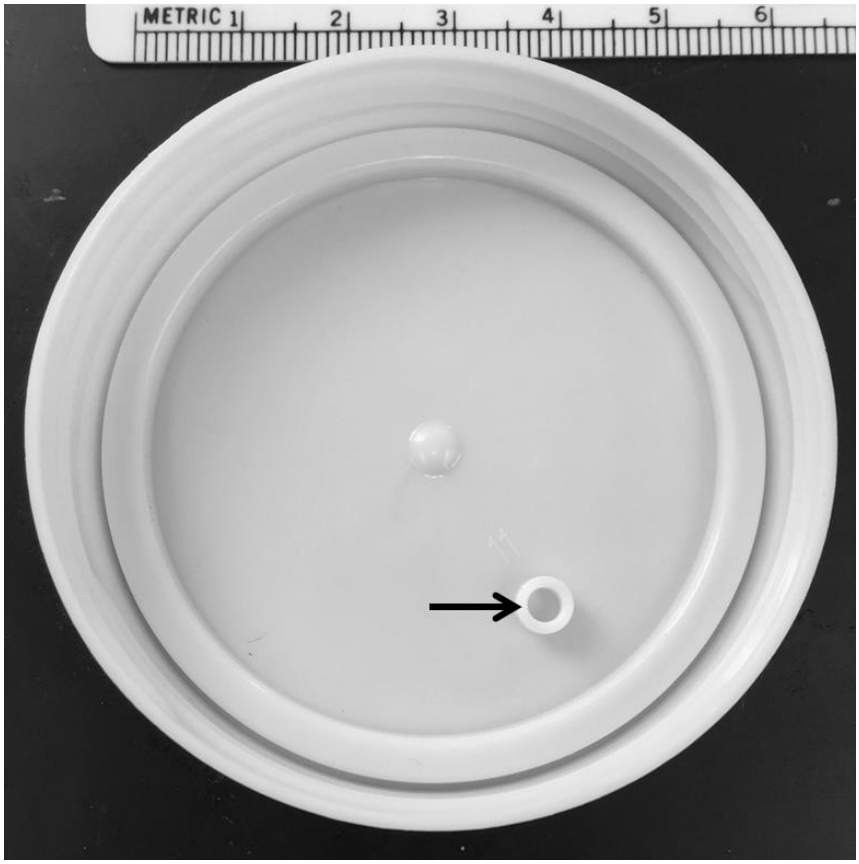

Figure S3. Photograph of the specimen cup lids used in the cercaria swimming speed experiment. Arrow indicates the depression used to contain individual cercaria during video capture. Metric ruler included for scale.

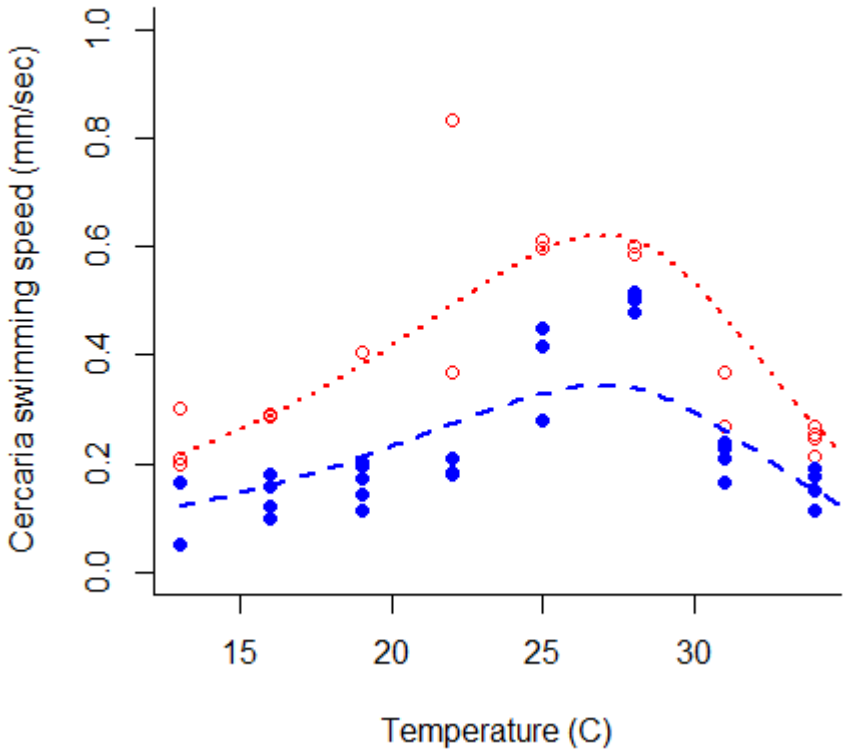

Figure S4. Visualization of the artificially created (fake) block effect on the *S. mansoni* cercariae swimming speed experimental data. Blue, closed circles represent swimming speeds below the mean swimming speed at that temperature (Block 1), and red, open circles represent swimming speed above the mean speed at that temperature (Block 2). The best-fitting SS-model that accounts for this block effect (Block 1: blue dashed curve; Block 2: red dotted curve) was estimated using the code in S4.5.

Literature Cited for Supporting Information:

Anderson, R. M. and R. M. May. 1978. Regulation and stability of host-parasite population interactions: I. Regulatory processes. Journal of Animal Ecology **47:** 219-247.

Anderson, R. M. and R. M. May. 1991. Infectious diseases of humans: Dynamics and control. Oxford Science Publications, Great Britain, 757 p.

Braner, M. and N. G. Hairston. 1989. From cohort data to life table parameters via stochastic modeling. *In* Estimation and analysis of insect populations, L. L. McDonald, B. F. J. Manly, J. A. Lockwood, and J. A. Logan (eds.). Springer-Verlag, Berlin, p. 81-92.

Brown, J. H., J. F. Gillooly, A. P. Allen, V. M. Savage and G. B. West. 2004. Toward a metabolic theory of ecology. Ecology **85:** 1771-1789.

Cohen, L. N., H. Neimark and L. K. Eveland. 1980. *Schistosoma mansoni*: response of cercariae to a thermal gradient. Journal of Parasitology **66:** 362-364.

Colley, D. G., A. L. Bustinduy, E. Secor and C. H. King. 2014. Human schistosomiasis. Lancet **383:** 2253-2264.

Edelstein, A. D., M. A. Tsuchida, N. Amodaj, H. Pinkard, R. D. Vale and N. Stuurman. 2014. Advanced methods of microscope control using μManager software. Journal of Biological Methods **1:** e11.

Gillooly, J. F., J. H. Brown, G. B. West, V. M. Savage and E. L. Charnov. 2001. Effects of size and temperature on metabolic rate. Science **293:** 2248-2251.

Hurford, A., D. Cownden and T. Day. 2010. Next-generation tools for evolutionary invasion analyses. Journal of the Royal Society Interface **7:** 561-571.

Keeling, M. J. and P. Rohani. 2008. Modeling infectious diseases in humans and animals. Princeton University Press, Princeton, N.J., 366 p.

LaFonte, B. E., T. R. Raffel, I. N. Monk and P. T. J. Johnson. 2015. Quantifying larval trematode infections in hosts: A comparison of method validity and implications for infection success. Experimental Parasitology **154:** 155-162.

Lawson, J. R. and R. A. Wilson. 1980. The survival of the cercariae of *Schistosoma mansoni* in relation to water temperature and glycogen utilization. Parasitology **81:** 337-348.

Molnár, P. K., S. J. Kutz, B. M. Hoar and A. P. Dobson. 2013. Metabolic approaches to understanding climate change impacts on seasonal host-macroparasite dynamics. Ecology Letters **16:** 9-21.

Nash, J. C. 2014. On best practice optimization methods in R. Journal of Statistical Software **60:** 1-14.

Nash, J. C. and R. Varadhan. 2011. Unifying optimization algorithms to aid software system users: optimx for R. Journal of Statistical Software **43:** 1-14.

Paull, S. H., T. R. Raffel, B. E. LaFonte and P. T. J. Johnson. 2015. How temperature shifts affect parasite production: testing the roles of thermal stress and acclimation. Functional Ecology **29:** 941-950.

R Development Core Team. 2014. R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria.

Raffel, T. R., J. M. Romansic, N. T. Halstead, T. A. McMahon, M. D. Venesky and J. R. Rohr. 2013. Disease and thermal acclimation in a more variable and unpredictable climate. Nature Climate Change **3:** 146-151.

Régnière, J., J. Powell, B. Bentz and V. Nealis. 2012. Effects of temperature on development, survival and reproduction of insects: Experimental design, data analysis and modeling. Journal of Insect Physiology **58:** 634-647.