

Novel Code

Individual growth rates and size at metamorphosis increase with watershed area in a salamander.

Cochrane, M. M., Addis, B. A., Swartz, L. K., and W. H. Lowe

Simulating Growth Data

To start simulating fake growth data we included 50 initial body sizes drawn from a random uniform distribution with a minimum of 35 mm and a maximum of 115 mm. We then randomly assigned between 5 and 400 days between all initial sizes and five subsequent capture events for each individual. To estimate growth over time we used the Von Bertalanffy (VB) growth function for one individual with at least two captures (Fabens 1965, Eaton and Link 2011):

$$S_t = S_{t-1} + (a - S_{t-1})(1 - e^{-k\Delta t})$$

Where S = size at capture, t = time step, a = asymptotic size, and k = growth rate.

We set $a = 120$ and $k = 0.20$ in order to simulate fake growth data over time for each individual. This allowed us to compare estimated model parameter values to the given ones above to ensure our model was correct.

We also simulated random watershed areas for all individuals with mean = 1.5 and SD = 1, in order to estimate the effect of watershed area on k .

```
# generate 50 initial sizes with min = 35, max = 115
S1 <- runif(n=50, min=35, max=117)

# generate random time steps (between 5-400 days) for all 50 individuals across 5 captures
t1 <- sample(5:400, 50, replace=FALSE)
t2<-sample(5:400, 50, replace=FALSE)
t3 <- sample(5:400, 50, replace=FALSE)
t4<-sample(5:400, 50, replace=FALSE)
t5 <- sample(5:400, 50, replace=FALSE)

# turn vectors for initial size and time steps into a dataframe
growth_data<-data.frame(t1,t2,t3,t4,t5,S1)

# set asymptotic size and growth rate parameter to simulate subsequent capture sizes
a = 120
k = 0.20

# generate size at capture for five additional capture occasions for all 50 individuals
growth_data <- growth_data %>%
  mutate(S2 = (S1 + (a-S1)*(1-exp(-k*(t1/365)))) %>%
  mutate(S3 = (S2 + (a-S2)*(1-exp(-k*(t2/365)))) %>%
  mutate(S4 = (S3 + (a-S3)*(1-exp(-k*(t3/365)))) %>%
  mutate(S5 = (S4 + (a-S4)*(1-exp(-k*(t4/365)))) %>%
```

```

mutate(S6 = (S5 + (a-S5)*(1-exp(-k*(t5/365))))))

# simulating fake watershed areas for all individuals
area.km2<-abs(rnorm(n=50, mean=1.5, sd=1))
# scale and center area
area <- as.vector(scale(area.km2))

# add watershed area to growth dataframe
growth_data$area.km2 <- area.km2
growth_data$area <- area

# view fake growth data and time steps
head(growth_data)

```

```

##      t1  t2  t3  t4  t5      S1      S2      S3      S4      S5      S6
## 1 111 386 131 148 260 63.60689 66.93460 77.05079 80.02567 83.13946 88.03391
## 2   64 136 102   43 158 37.10116 39.95791 45.70586 49.74429 51.38028 57.07116
## 3 145   81 341 361 267 50.23883 55.56702 58.36425 68.86891 78.04555 83.75567
## 4   55 205 284 265   95 86.75394 87.74093 91.16845 95.32344 98.65861 99.74111
## 5   47 212 232 219 263 89.81921 90.58655 93.81235 96.93850 99.54628 102.29132
## 6   69 124 109 129 290 84.76575 86.07302 88.30163 90.13942 92.17724 96.26501
##      area.km2      area
## 1 2.7587096 1.03896995
## 2 0.8165623 -1.16778835
## 3 1.4195381 -0.48265913
## 4 1.8891603 0.05094753
## 5 3.0458879 1.36527536
## 6 2.0467163 0.22996995

```

```

# separate out encounter history
y <- as.matrix(growth_data[,6:11])

# separate out days since last capture
dslc <- as.matrix(growth_data[,1:5])

# set number of individuals
NIND <- nrow(growth_data)

```

Organizing Data to Run Model in JAGS

We fit a Bayesian VB growth model using Markov chain Monte Carlo (MCMC) methods in JAGS (Plummer 2003).

```

# format data for jags

# NIND = number of individuals
# y = SVL (size) measurements
# f = first capture occasion
# l = last capture occasion

### functions for data manipulation
# determine the occasion of the first capture of each individual

```

```

get.first <- function(x){
  # a function to identify the time period of the first capture from an encounter history matrix
  return( min(which( x != 0 ) ) )
}

# get last occasion of capture for each individual
get.last <- function(x){
  # a function to identify the time period of the first capture from an encounter history matrix
  return( max(which( x != 0 ) ) )
}

fall <- apply( y, 1, get.first )
f <- fall # vector of first cap time step

lall <- apply(y, 1 ,get.last)
l <- lall # vector for which time period has last capture

### get ready to run JAGS
# Define a list of data to be passed to JAGS
growth.data <- list( y = y, NIND = NIND, f= f, l = l, dslc = dslc, area=area)

# set parameters (measurement error, asymptotic size,
# transformed growth rate parameter, effect of area on k) to track in JAGS
growth.parms <- c("tau.eps", "a", "mean.k", "area.k")

# set initial values for parameters
sal.inits <- function(){
  list(
    tau.eps = runif( 1, 0, 1 ),
    mean.k = runif( 1, -5, 5 ),
    a = runif( 1, 80, 130 ),
    area.k = runif( 1, -5, 5 )
  )
}

# set up for MCMC run
ni <- 5000 # number of iterations
nt <- 5 # thinning number, e.g. only keeping out of 5 values
nb <- 3000 # burn in
nc <- 3 # number of chains
# total iterations kept = (ni/nb)/nt*nc

```

JAGS Model Script

To run our Bayesian VB growth model we set priors and defined the likelihood below. This initial (simplified) model includes the linear effect of watershed area on k . For this initial model, k is calculated across the entire year equally. We also estimate a parameter, τ_{eps} , that accounts for measurement error.

```

model {
  # Prior Specification (Growth Model)

```

```

a ~ dunif(80,130) # uniform prior for asymptotic size with minimum of 80 and maximum of 130

tau.eps ~ dgamma(0.001, 0.001)
# gamma prior for measurement error with shape = 0.001, rate = 0.001

sd.eps <- 1/sqrt(tau.eps) # measurement error sd

mean.k ~ dunif(-12,0)
# coefficient for the log of the active growth rate parameter (limits k between 0 and 1)

area.k ~ dnorm(0,0.01) # normal prior for the effect of area on mean k

# Likelihood
for (i in 1:NIND){ # Loop through individuals
  for (t in f[i]:f[i]){ # First survey

    y[i,t] ~ dnorm(H[i,t],tau.eps) # Measured size is the true size +/- measurement error

    H[i,t] ~ dunif(30,120) # Prior: Expected size at first capture (between 30 and 120)
  }

  for (t in (f[i]+1):l[i]){ # Loop through surveys
    # (1st capture to last survey date for each individual)

    y[i,t] ~ dnorm(H[i,t],tau.eps) # measured size is a function of
    # true size plus measurement error

    # Expected size at time t = H

    H[i,t] <- H[i,t-1]+(a-H[i,t-1])*

      (1-exp(-k[i,t]*(dslc[i,t-1]/365)))

    log(k[i,t]) <- mean.k + area.k*area[i]

    # log-link function to allow active k to vary in relation to watershed area

  }
}
}

```

Run Model and View Results

We used the package R2jags (Su and Yajima 2021) to run our JAGS model, and the package bayesplot (Gabry and Mahr 2022) to check for model convergence and view posterior estimates, all in RStudio (Version 1.4.1716; R Development Core Team 2021).

Posterior estimates for a should be around 120 and estimates for k should be around 0.20 if model is set up correctly.

```
setwd("C:/Users/maddy/Box Sync/HB/Data/growth/")
```

```
# run model in R2jags
```

```
vbgm.result.area <- R2jags::jags(data=growth.data,
                                inits = sal.inits,
                                parameters.to.save=growth.parms,
                                model.file="vbgm/jags/areamodel.txt",
                                n.chains=nc,
                                n.iter=ni,
                                n.burnin=nb,
                                n.thin=nt)
```

```
## module glm loaded
```

```
## Compiling model graph
```

```
##   Resolving undeclared variables
```

```
##   Allocating nodes
```

```
## Graph information:
```

```
##   Observed stochastic nodes: 300
```

```
##   Unobserved stochastic nodes: 54
```

```
##   Total graph size: 2649
```

```
##
```

```
## Initializing model
```

```
# view summary table
```

```
vbgm.result.area$BUGSoutput$summary
```

##	mean	sd	2.5%	25%	50%
## a	1.200004e+02	2.347093e-03	1.199958e+02	1.199988e+02	1.200004e+02
## area.k	-1.922942e-06	2.000685e-05	-4.094781e-05	-1.633157e-05	-1.814509e-06
## deviance	-2.911920e+03	3.023906e+01	-2.968689e+03	-2.931592e+03	-2.911226e+03
## mean.k	-1.609448e+00	5.514191e-05	-1.609555e+00	-1.609485e+00	-1.609447e+00
## tau.eps	1.238039e+05	1.151888e+04	1.025645e+05	1.155926e+05	1.231303e+05
##	75%	97.5%	Rhat	n.eff	
## a	1.200020e+02	1.200048e+02	1.000000	1	
## area.k	1.224272e-05	3.787650e-05	1.000000	1	
## deviance	-2.892873e+03	-2.849513e+03	1.000188	1200	
## mean.k	-1.609411e+00	-1.609343e+00	1.000000	1	
## tau.eps	1.316459e+05	1.468164e+05	1.000146	1200	

```
# a parameter estimate should be ~ 120
```

```
# area.k parameter estimate should be ~ 0
```

```
# we want all Rhat values < 1.05
```

```
# get an un-transformed estimate of k (should be ~ 0.20 if model worked)
```

```
mean.k<-exp(vbgm.result.area$BUGSoutput$sims.list$mean.k)
```

```
quantile(mean.k, 0.50) # mean k
```

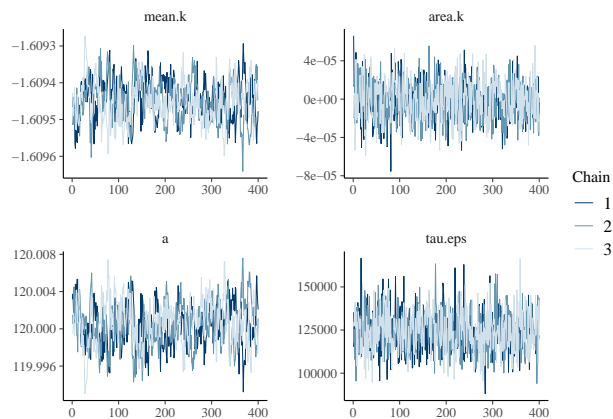
```
##           50%
```

```
## 0.1999981
```

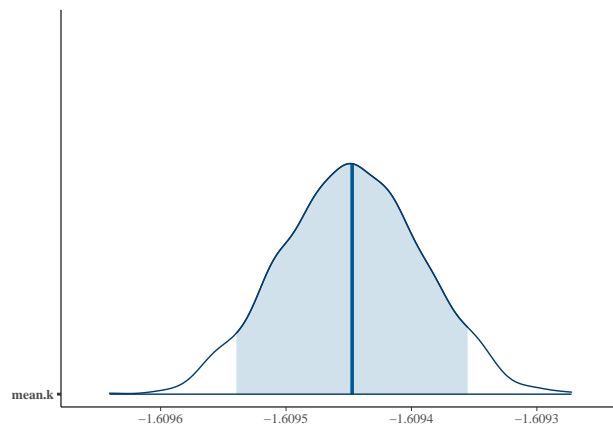
```
# standard deviation of measurement error
tau.eps<-exp(vbgm.result.area$BUGSoutput$sims.list$tau.eps)
mean(1/sqrt(tau.eps)) # should be pretty close to 0
```

```
## [1] 0
```

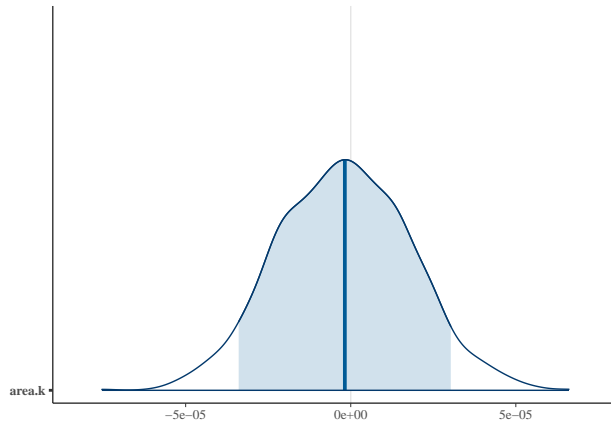
```
# view traceplots to see if chains converged
posterior <- as.array(vbgm.result.area$BUGSoutput$sims.array)
bayesplot::mcmc_trace(posterior, pars = c("mean.k", "area.k", "a", "tau.eps"))
```



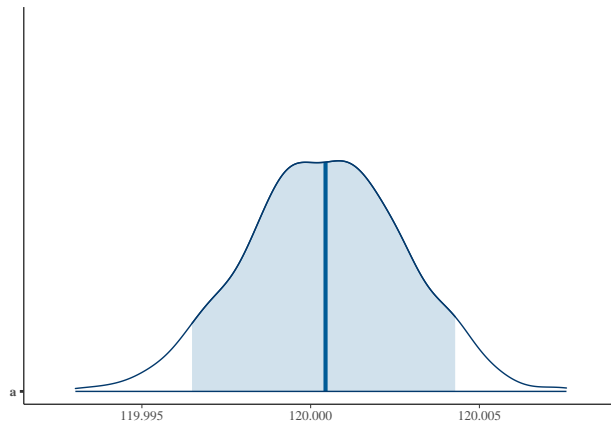
```
# view posterior uncertainty intervals
mcmc_areas(posterior,
  pars = c("mean.k"),
  prob = 0.9) # 90% credible interval
```



```
mcmc_areas(posterior,
  pars = c("area.k"),
  prob = 0.9) # 90% credible interval
```



```
mcmc_areas(posterior,
  pars = c("a"),
  prob = 0.9) # 90% credible interval
```



Model Validation

We calculated a Bayesian p-value to make sure our observed data matched our model output.

```
#### posterior predictive check ####
# comparing observed data to posterior predictive distribution
# bayesian p -value = probability that the replicated data could be
# more extreme than the observed data

# posterior distributions
a_output <- vbgm.result.area$BUGSoutput$sims.list$a
area_k_output <- vbgm.result.area$BUGSoutput$sims.list$area.k
mean_k_output <- vbgm.result.area$BUGSoutput$sims.list$mean.k

# create data frame with posterior distributions for all parameters
mod_output <- data.frame(a_output, area_k_output, mean_k_output)

# add in first time step SVL and time to next capture, in addition to discharge co-variates on k
mod_output$original_SVL <- growth_data$S1
```

```

mod_output$days <- growth_data$t1
mod_output$area <- growth_data$area

# calculate size at next time step based on posterior iterations
mod_output <- mod_output %>%
  mutate(k = exp(mean_k_output + area_k_output*area )) %>%
  mutate(SVL2_model = original_SVL + (a_output - original_SVL)*(1-exp((-k*(days/365))))))

# add in original size at next time step to df
mod_output$SVL2_orig <- growth_data$S2

head(mod_output)

```

```

##   a_output area_k_output mean_k_output original_SVL days      area      k
## 1 120.0009 2.982214e-06   -1.609433    63.60689  111  1.03896995 0.2000015
## 2 120.0017 2.233307e-05   -1.609477    37.10116   64 -1.16778835 0.1999869
## 3 120.0008 2.236232e-05   -1.609484    50.23883  145 -0.48265913 0.1999886
## 4 119.9989 -1.932822e-06   -1.609432    86.75394   55  0.05094753 0.2000011
## 5 119.9986 -1.896531e-05   -1.609384    89.81921   47  1.36527536 0.2000055
## 6 119.9948 2.046660e-06   -1.609343    84.76575   69  0.22996995 0.2000191
##   SVL2_model SVL2_orig
## 1   66.93468  66.93460
## 2   39.95779  39.95791
## 3   55.56679  55.56702
## 4   87.74090  87.74093
## 5   90.58653  90.58655
## 6   86.07295  86.07302

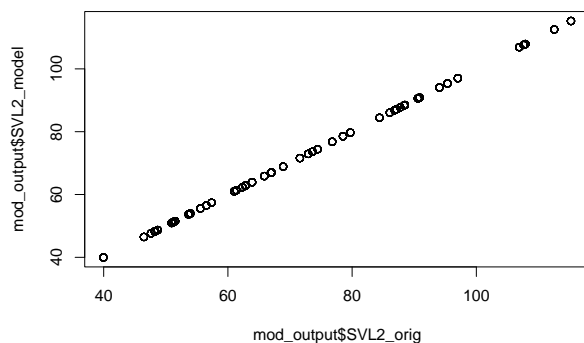
```

```

# calculate the proportion of simulated values that are equal to or greater than the original dataset
mod_output <- mod_output %>%
  mutate(greater_than = if_else(SVL2_model >= SVL2_orig, 1,0))

# plot original size compared to modeled size
# want all points on or near 1:1 line
plot(mod_output$SVL2_orig, mod_output$SVL2_model)

```




```
# calculate bayesian p-value
sums<-mod_output%>%
  dplyr::count(greater_than)
sums
```

```
##   greater_than    n
## 1             0 584
## 2             1 616
```

```
sums$n[2]/1200
```

```
## [1] 0.5133333
```

```
# bayesian p-value should be close to 0.5 in order to validate the model
```

Plot Growth Trajectory from Parameter Estimates

In order to plot growth over time based on posterior parameter estimates we also had to define the initial size of animals as 14 mm based on historic observations. In order to do this we used a different organization of the VB growth function to plot growth over time:

$$S_t = a(1 - be^{-kt})$$

Where S_t = size at age t , a = asymptotic size estimated from model, hatchling size = $a(1-b)$, and k = growth rate parameter estimated from model.

```
# set initial size at time 0
init.size <- 14

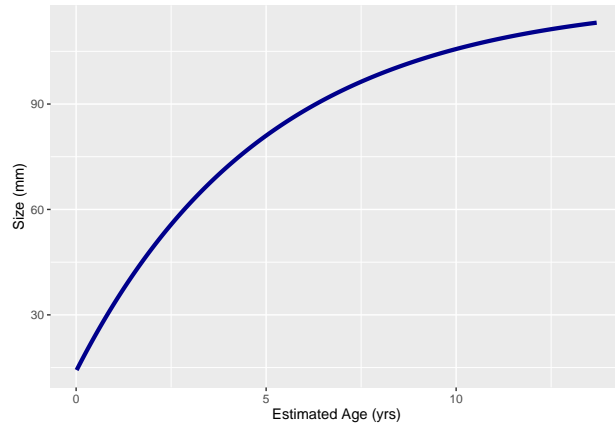
# get parameter estimates from model for a and k
mean.a<-vbgm.result.area$BUGSoutput$mean$a
mean.k <-vbgm.result.area$BUGSoutput$mean$mean.k

# create a dataframe with a column for days since hatching
days_vec <- seq(5,5000,by=1)
growth_traj <- data.frame(days_vec)

# project growth across each time step given parameter estimates
growth_traj <- growth_traj%>%
  mutate(k =exp(mean.k))%>%
  mutate(a = mean.a)%>%
  mutate(init.size = init.size)%>%
  mutate(size = init.size + (a-init.size)*(1-exp(-k*(days_vec/365))))

# plot size on y axis and age on x axis
ggplot(data = growth_traj, aes(x=days_vec/365, y=size)) +
  # set x axis to yrs instead of days
  geom_line(size=1.5, color="darkblue")+
  labs(y="Size (mm)", x = "Estimated Age (yrs)")
```

```
## Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use `linewidth` instead.
```



Final JAGS Model Script

Below is the script for our final VB growth model, which includes added complexity (compared to the model described above) when estimating parameters to describe the effect of covariates on k . For example, this model includes the relationship between k and watershed area, in addition to the effect of fish presence and stream on k , and separates out active season verse inactive season growth rates.

```
model {
  # Prior Specification
  a ~ dunif(80,130) # uniform prior on asymptotic size, min = 80, max = 130

  tau.eps ~ dgamma(0.001, 0.001) # gamma prior on measurement error with
  # shape = 0.001, rate = 0.001

  sd.eps <- 1/sqrt(tau.eps) # measurement error sd

  mean.k.active ~ dunif(-12,0)
  # coefficient for the log of the active growth rate parameter (limits k between 0 and 1)

  area.k.active ~ dnorm(0,0.01) # normal prior on linear effect of area on mean k during summer

  fish.k.active ~ dnorm(0,0.01) # normal prior on the effect of fish on k

  for(s in 1:STREAMS){
    k[s] ~ dnorm(0,0.01) # allowing k to vary by stream name
  }

  mean.k.inactive ~ dunif(-12,0)
  # coefficient for the log of the active growth rate parameter (limits k between 0 and 1)

  # Likelihood
  for (i in 1:NIND){ # Loop through individuals

    for (t in f[i]:f[i]){ # First survey

      y[i,t] ~ dnorm(H[i,t],tau.eps) # Measured size is the true size +/- measurement error

      H[i,t] ~ dunif(30,120) # Prior: Expected size at first capture
    }
  }
}
```

```

}

for (t in (f[i]+1):l[i]){ # Loop through surveys (1st capture to last
  # survey date for each individual)

  y[i,t] ~ dnorm(H[i,t],tau.eps) # measured size is a function of true size plus measurement error

  # Expected size at time t = H
  H[i,t] <- H[i,t-1]+(a-H[i,t-1])*
    (1-exp(-k.active[i,t]*(dslc_active[i,t-1]/365)- k.inactive[i,t]*(dslc_inactive[i,t-1]/365)))

  # log-link function to allow active k to vary in relation to area, fish
  log(k.active[i,t]) <- mean.k.active + area.k.active*area[i] +
    fish.k.active*fish[i] + k[stream[i]]

  log(k.inactive[i,t]) <- mean.k.inactive
}}

```

Literature Cited

- Eaton, M. J., and W. A. Link. 2011. Estimating age from recapture data: Integrating incremental growth measures with ancillary data to infer age-at-length. *Ecological Applications* 21:2487–2497.
- Fabens, A. J. 1965. Properties and fitting of the von Bertalanfy growth curve. *Growth* 29:265–289.
- Gabry J, and T. Mahr. 2022. “bayesplot: Plotting for Bayesian Models.” R package version 1.9.0.
- Plummer, M. 2003. JAGS: A program for analysis of Bayesian graphical models using Gibbs sampling. *Proceedings of the 3rd international workshop on distributed statistical computing* 125:1–10.
- Su, Y. S., and Yajima, M. 2021. R Package “R2jags: A Package for Running jags from R”.