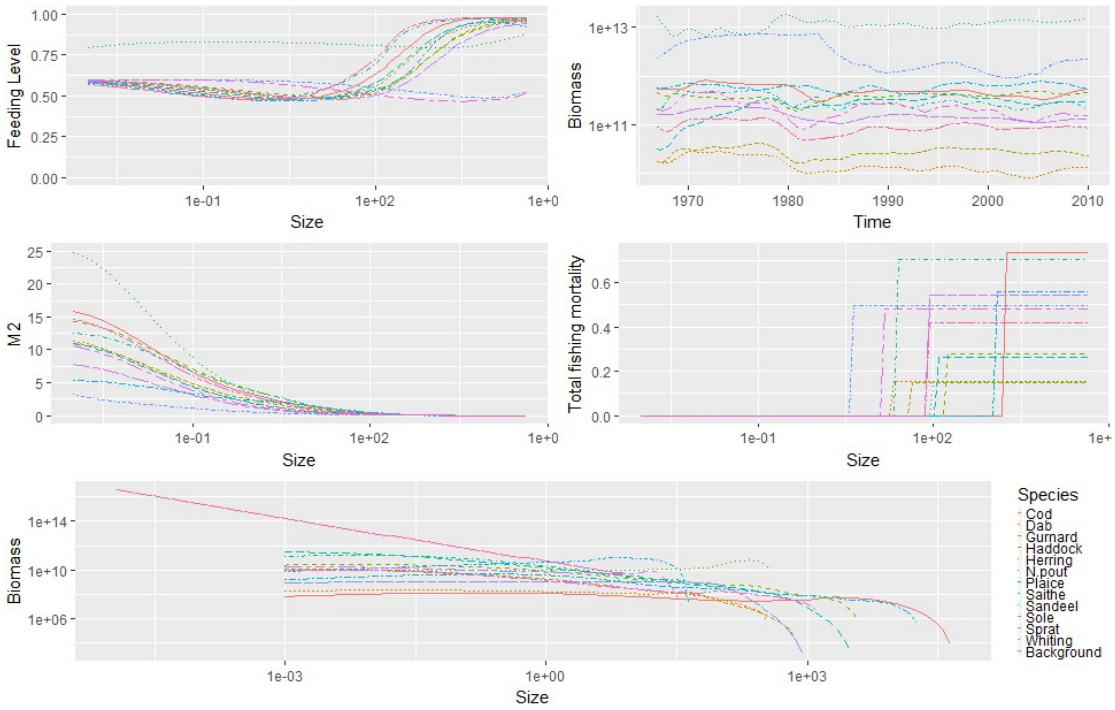
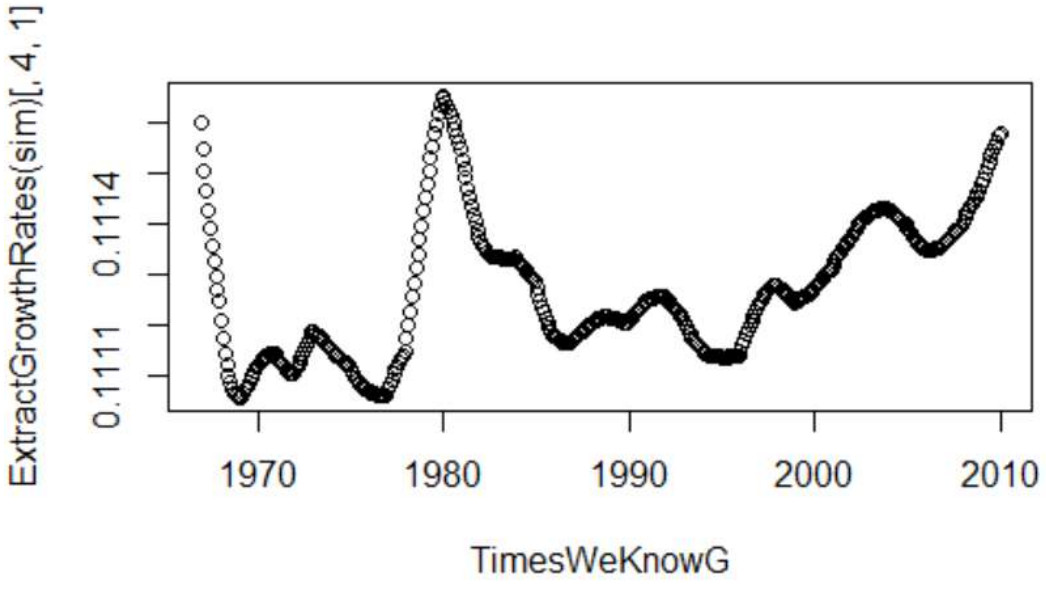
Plots for growth parameter fitting project - Richard Southwell

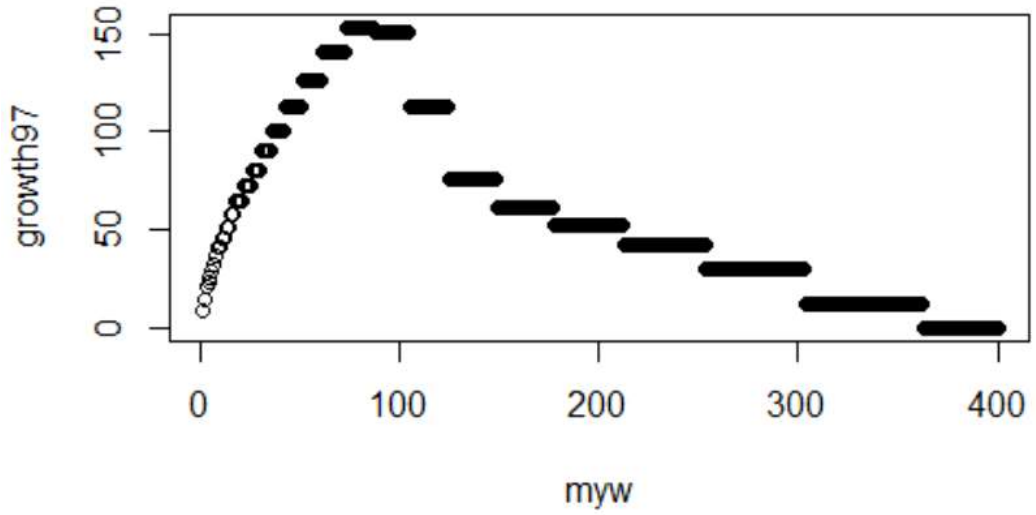
sim



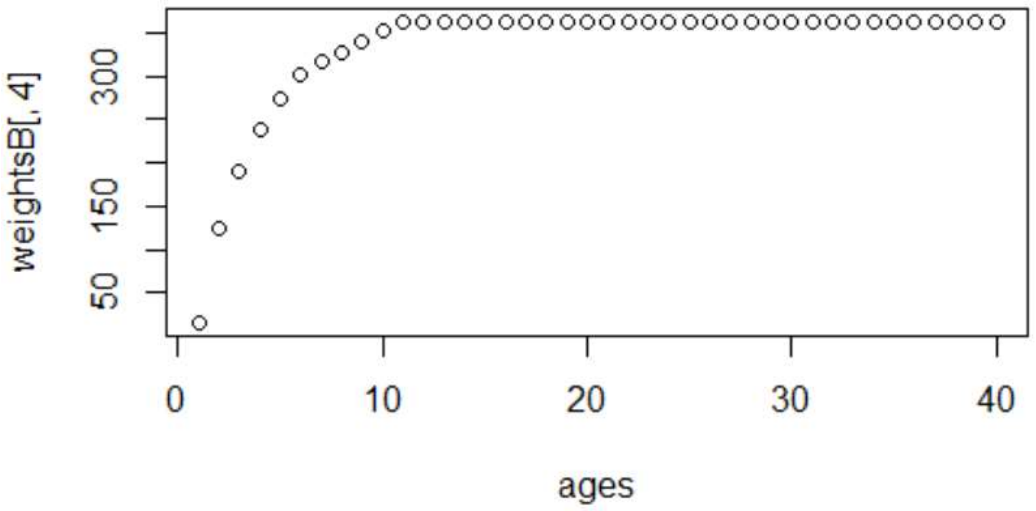
growthrates



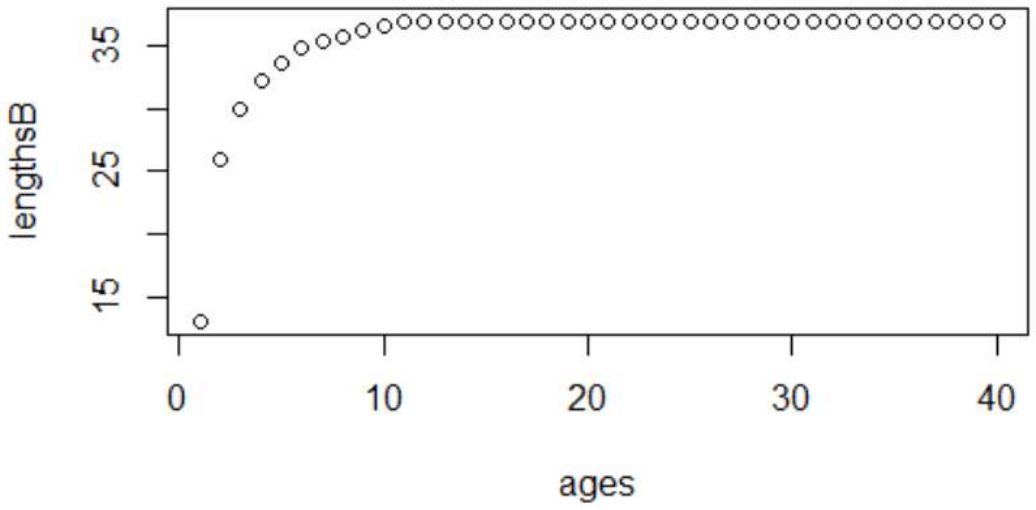
growth97



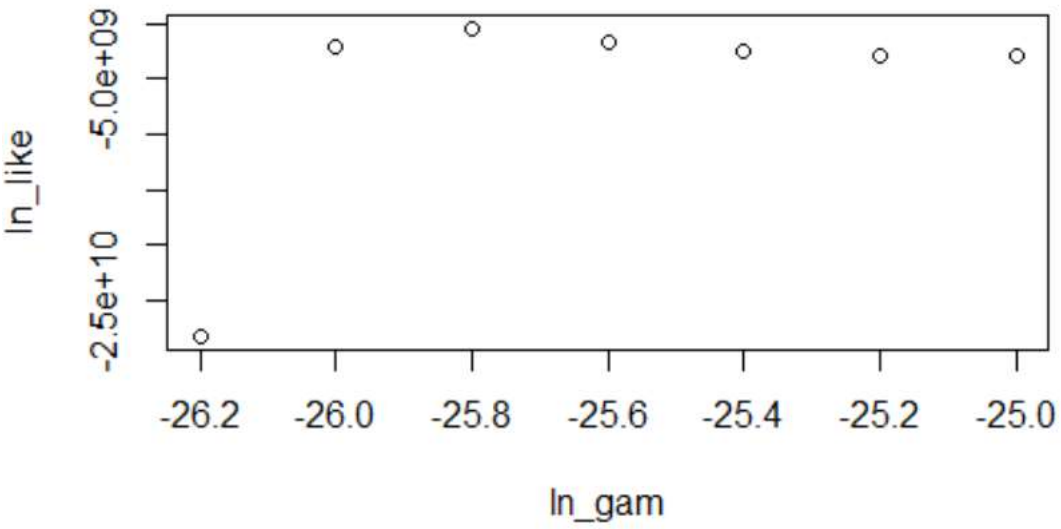
ageVweight



ageVlength



loglike



loglikevals

```
>
> c(best_log_gam,exp(best_log_gam),max(ln_like))
[1] -2.580000e+01  6.240255e-12 -3.732351e+08
> #-2.590000e+01  5.646417e-12 -1.252503e+07
>
> log(param1fst@species_params[4,15])
[1] -25.21224
> # -25.21224
> plot(ages,weightsB[,4])
```

Pictures are refered to in [square brackets] in code comments below:

%%%%%%%% code %%%%%%%% dyngrowth.R%%%%%%%%%%%

set.seed(5)

library(devtools)

library(ggplot2)

library(grid)

library(methods)

library(plyr)

library(reshape2)

library(mizer)

library(FME)

library(reshape2)

library(deSolve)

library("plot3D")

library(rgl)

library("plot3Drgl")

library(optimx)

library(mvtnorm)

```
# Run appropriate core mizer files again, that have been
# rewritten to include sim@growth slots
source("R/project.R")
source("R/MizerSim-class.R")

# Fmat is fishing effort data, we use it from 1967 until 2010
# for our mizer runs
load("Fmat.RData")

# Load basic North sea parameters
params_data <- read.csv("./vignettes/NS_species_params.csv")
inter <- read.csv("./vignettes/inter.csv", row.names=1)
inter <- as(inter, "matrix")

# Set Fmat be be over the proper years
Fmat <- Fmat[,11:(dim(Fmat)[2]-1)]

# a and b are such that W=a*L^b for Herring (species # 4)
# TO DO: get these a's and b's for other NS species, and generalize
a <- 0.006
b <- 3.05

# This loads mike's samples from posterior of VB parameters, given
# some empirical data
```

```
load("Herring.Rdata")

#out is name of this data = k, Linf,

out2 <- out

out2[,2] <- out[,2]/10


# We these samples from mike's distribution and find the mean MU and covariance SIGMA

# of these samples, then we use the resulting gaussian as an approx of the

# underlying distribution mike sampled from

SIGMA<-cov(out2[,1:2])

MU<-colMeans(out2[,1:2])


# gamma is the volumetric search rate constant (for Herring)

# set the log gamma value to work with.

loggam <- -24

# mizer-encoded `empirical` value is -25.21224

# $$$$$$$$ # loglikk <- function(loggam){


# Initialize mizer with altered gamma

param1 <- MizerParams(params_data, interaction = inter, no_w = 100)

wG <- log(param1@species_params[,15])

wG[4] <- loggam

params_dataB <- params_data

## params_dataB[["h"]] <- param1@species_params[,14]

params_dataB[["gamma"]] <- exp(wG)

params_dataB[["w_inf"]] <- params_data[["w_inf"]]

params <- MizerParams(params_dataB, interaction = inter, no_w = 100)
```

```
# run once over 1967-2010 to make a good initial condition

# (it looks like it is periodic)

primer <- project(params, effort = t(Fmat), dt = 0.1, t_save =.1)

sim <- project(params, effort = t(Fmat), dt = 0.1, t_save =.1,initial_n=
primer@n[nrow(primer@n),,], initial_n_pp=primer@n_pp[nrow(primer@n_pp),])


# [sim] picture

plot(sim)


# get the saved growth rates, and rewrite the growth rates on

# the last step correctly (by default they are same as on penultimate

# step)


# Here ExtractGrowthRates(sim)[k,i,t] equals

# the growth rate g(W,T) where W=params@w[k] and

# T = as.numeric(rownames(sim@n_pp))[t]

# is the t th time point at which mizer saves output



ExtractGrowthRates <- function(sim){

  HH <- sim@growth

  HH[dim(HH)[1],,] <-
getEGrowth(params,sim@n[nrow(sim@n),,],sim@n_pp[nrow(sim@n_pp),])

  return(HH)

}
```

```
# Get the times we  run mizer over, and w
TimesWeKnowG <- as.numeric(rownames(sim@n_pp))
w_pts <- params@w

#growth rates of small herring
#[growthrates]
plot(TimesWeKnowG, ExtractGrowthRates(sim)[,4,1])



# For interpolation between w and t that we have g(w,t) for
# I use these functions that return greatest index of a point
# with a value <= input
get_w_index <- function(W){
  return(length(w_pts[w_pts <=W]))
}
get_t_index <- function(T){
  return(length(TimesWeKnowG[TimesWeKnowG<=T]))
}
# TO DO: make these indexers better, so any input
# returns a usable index-output (possibly with warning msg)

# eg is the extracted growth rates
eg <- ExtractGrowthRates(sim)
```

```
# Ginter(W,T,eg) uses our interpolation to get
# any growth rates required during ODE solution

Ginter <- function(W,T,eg){
  return(eg[get_t_index(T),,get_w_index(W)])
}

# plot growth rates of herring in 1997
# [growth97]
myw <- (1:400)
growth97 <- sapply(myw, function(x) Ginter(x,1997.734,eg)[4])
plot(myw,growth97)


# solve ODE to get age vs length data
# for a Herring born in 1976
###EarlyBirthData <- matrix(0,nrow=length(TimesWeKnowG),ncol=12)
ages <- seq(1,40,1)
EarlyBirthData <- matrix(0,nrow=length(ages),ncol=12)
for (i in (1:12)){
  myodefun <- function(t, state, parameters){
    return(list(Ginter(state,t,eg)[i]))
  }
  ageWeightGenBirth <- function(t0) {
    # for some reason lsoda gives errors, so we use euler for now
    ## return(ode(y = w_pts[1],
```

```
  ##          times = TimesWeKnowG[TimesWeKnowG>=t0], func = myodefun, parms = 1))
  # TO DO: Understand and fix this
  return(ode(y = w_pts[1],
          times = TimesWeKnowG[TimesWeKnowG>=t0], func = myodefun, parms = 1,
method = "euler"))
 }
 ###EarlyBirthData[,i] <- ageWeightGenBirth(TimesWeKnowG[1])[,2]
 weightss <- ages
 for (m in (1:length(ages))){
  XQ <- ageWeightGenBirth(TimesWeKnowG[length(TimesWeKnowG)]-ages[m])
  weightss[m] <- XQ[dim(XQ)[1],2]
 }
 EarlyBirthData[,i] <- weightss
}
weightsB <- EarlyBirthData
# TO DO *:  Cross check how VB works wrt t0

# TO DO: Speed up this iterpolation and ODE solving

# TO DO: Use linear interpolation

# TO DO: improve VB parameter fitting method

# [ageVweight]
plot(ages,weightsB[,4])
```

```
# convert weights to lengths
lengthsB <- sapply(weightsB[,4],function(x) exp(log(x/a)/b))

# [ageVlength]
plot(ages,lengthsB)



# do nonlinear least squares fitting to find VB parameters best
# fitting mizer's output
###datsB <- data.frame(X=TimesWeKnowG-TimesWeKnowG[1], Y= lengthsB)
datsB <- data.frame(X=ages, Y= lengthsB)
param1 <- params
W_egg <- param1@w[1]
L_egg <- (W_egg/a)^(1/b)
gett0 <- function(Linf,k){
  return(log(1-L_egg/Linf)/k)
}
vbTyp<-function(X,Linf,k){(Linf*(1-exp(-k*(X-gett0(Linf,k)))))}
obs_k <- MU[[1]]
obs_Linf <- MU[[2]]
fitTypB<-nls(Y~vbTyp(X,Linf,k),data=datsB,start=list(Linf=obs_Linf,k=obs_k))
vbfitB <- coef(fitTypB)

# Compute likelihood, which is prob of best fitting VB parameters
# with respect to the gaussian approximation of the
```

```
# posterior distribution mike sampled from

loglikeB <- dmvnorm(c(vbfitB[["k"]],vbfitB[["Linf"]]),MU,SIGMA,log=T)

loglikeB

#$$$$$$$$$$$$$$$$$$

#return(loglikeB)}

#[loglike]

#[loglikevals]
```