Note: this version still uses the old ode\_rk45 functions (integrate\_ode\_rk45)

still working as of rstan Z.21.2 but deprecated in cmdstanr Z.24

###########################################################################

# Project : Some test of ODE in R from

# https://www.magesblog.com/post/2021-02-08-fitting-multivariate-ode-models-with-brms/

# Developed in : R version 4.0.3 (2020-10-10)

# Purpose : Test ODE in brms with cmdstan

# Revision History :

# Version Date Author Revision

# ------- --------- ---------- -------------------------

# 1.0 23-Feb-2021 Pierre Lebrun Creation/copy-paste from blog

#

# Declaration of Confidentiality (choose the applicable sentence):

# - I, as an author, certify that this program is free from any client or

# client project information whatever the type of information and that

# this program can be used without risk of confidentiality issue.

#

###########################################################################

########################## Clean all ######################################

rm(list = ls())

########################## Libraries and wd ###############################

library(rstan)

options(mc.cores = parallel::detectCores())

rstan\_options(auto\_write = TRUE)

library(brms)

library(parallel)

nCores <- detectCores()

# we recommend running this is a fresh R session or restarting your current session

#install.packages("cmdstanr", repos = c("https://mc-stan.org/r-packages/", getOption("repos")))

library(cmdstanr)

#install\_cmdstan(cores = nCores)

#check\_cmdstan\_toolchain()

library(data.table)

library(tidybayes)

########################## Data import ####################################

Lynx\_Hare <- data.table(

Year = 1900:1920,

Lynx = c(4, 6.1, 9.8, 35.2, 59.4, 41.7, 19, 13, 8.3, 9.1, 7.4, 8,

12.3, 19.5, 45.7, 51.1, 29.7, 15.8, 9.7, 10.1, 8.6),

Hare = c(30, 47.2, 70.2, 77.4, 36.3, 20.6, 18.1, 21.4, 22, 25.4,

27.1, 40.3, 57, 76.6, 52.3, 19.5, 11.2, 7.6, 14.6, 16.2, 24.7))

head(Lynx\_Hare)

########################## Data management ################################

LH <- melt(data.table(Lynx\_Hare), id.vars = "Year",

measure.vars = c("Lynx", "Hare"),

variable.name = "Population",

value.name = "Pelts")

LH[, `:=`(

delta = ifelse(Population %in% "Lynx", 1, 0),

Population = factor(Population),

t = Year - min(Year) + 1)]

head(LH)

########################## Data graphs ####################################

ggplot(LH,aes(x = Year,y = Pelts,color = Population)) + geom\_line()

########################## Model(s) #######################################

LotkaVolterra <- "

// Sepcify dynamical system (ODEs)

real[] ode\_LV(real t, real [] y, real [] theta,

real [] x\_r, int[] x\_i){

real dydt[2];

dydt[1] = (theta[1] - theta[2] \* y[2] ) \* y[1]; // Hare

dydt[2] = (theta[3] \* y[1] - theta[4]) \* y[2]; // Lynx

return dydt;

}

// Integrate ODEs and prepare output

real LV(real t, real Hare0, real Lynx0,

real brHare, real mrHare,

real brLynx, real mrLynx,

real delta){

real y0[2]; // Initial values

real theta[4]; // Parameters

real y[1, 2]; // ODE solution

// Set initial values

y0[1] = Hare0; y0[2] = Lynx0;

// Set parameters

theta[1] = brHare; theta[2] = mrHare;

theta[3] = brLynx; theta[4] = mrLynx;

// Solve ODEs

y = integrate\_ode\_rk45(ode\_LV,

y0, 0, rep\_array(t, 1), theta,

rep\_array(0.0, 0), rep\_array(1, 1),

0.001, 0.001, 100); // tolerances, steps

// Return relevant population values

return (y[1,1] \* (1 - delta) +

y[1,2] \* delta);

}

"

frml <- bf(

Pelts ~ eta,

nlf(eta ~ log(

LV(t, Hare0, Lynx0,

brHare, mrHare,brLynx, mrLynx, delta)

)

),

nlf(Hare0 ~ 10 \* exp(stdNHare0)),

nlf(Lynx0 ~ 10 \* exp(stdNLynx0)),

nlf(brHare ~ 0.5 \* exp(0.25 \* stdNbrHare)),

nlf(mrHare ~ 0.025 \* exp(0.25 \* stdNmrHare)),

nlf(brLynx ~ 0.025 \* exp(0.25 \* stdNbrLynx)),

nlf(mrLynx ~ 0.8 \* exp(0.25 \* stdNmrLynx)),

stdNHare0 ~ 1, stdNLynx0 ~ 1,

stdNbrHare ~ 1, stdNmrHare ~ 1,

stdNbrLynx ~ 1, stdNmrLynx ~ 1,

sigma ~ 0 + Population,

nl = TRUE)

mypriors <- c(

prior(normal(0, 1), nlpar = "stdNHare0"),

prior(normal(0, 1), nlpar = "stdNLynx0"),

prior(normal(0, 1), nlpar = "stdNbrHare"),

prior(normal(0, 1), nlpar = "stdNmrHare"),

prior(normal(0, 1), nlpar = "stdNbrLynx"),

prior(normal(0, 1), nlpar = "stdNmrLynx"),

prior(normal(-1, 0.5), class = "b",

coef = "PopulationHare", dpar = "sigma"),

prior(normal(-1, 0.5), class = "b",

coef = "PopulationLynx", dpar = "sigma")

)

mod <- brm(

frml, prior = mypriors,

stanvars = stanvar(scode = LotkaVolterra, block = "functions"),

data = LH, backend = "cmdstan",

family = brmsfamily("lognormal", link\_sigma = "log"),

control = list(adapt\_delta = 0.99),

seed = 1234, iter = 1000,

chains = 4, cores = nCores,

file = "LotkaVolterraCMDStan.rds")

########################## Model Check ####################################

summary(mod)

theme\_update(text = element\_text(family = "sans"))

plot(mod, N = 8)

trnf <- function(m, par, a, b){

x <- unlist(posterior\_samples(m, par))

tx <- a \* exp(b \* x)

c(mean = mean(tx),

se\_mean = sd(tx)/sqrt(length(tx)),

sd = sd(tx),

quantile(tx, probs = c(0.025, 0.1, 0.5, 0.9, 0.975)))

}

round(rbind(

Hare0 = trnf(mod, 'stdNHare0\_Intercept', 10, 1),

Lynx0 = trnf(mod, 'stdNLynx0\_Intercept', 10, 1),

brHare = trnf(mod, 'stdNbrHare\_Intercept', 0.5, 0.25),

mrHare = trnf(mod, 'stdNmrHare\_Intercept', 0.025, 0.25),

brLynx = trnf(mod, 'stdNbrLynx\_Intercept', 0.025, 0.25),

mrLynx = trnf(mod, 'stdNmrLynx\_Intercept', 0.8, 0.25),

SigmaHare = trnf(mod, 'sigma\_PopulationHare', 1, 1),

SigmaLynx = trnf(mod, 'sigma\_PopulationLynx', 1, 1)

), 3)

########################## Prediction #####################################

# modRstan <- brm(

# frml, prior = mypriors,

# stanvars = stanvar(scode = LotkaVolterra, block = "functions"),

# data = LH, backend = "rstan", chains = 0, # don't run sims

# family = brmsfamily("lognormal", link\_sigma = "log"),

# file = "LotkaVolterraRStan")

expose\_functions(mod, vectorize = TRUE, cacheDir = "~/Downloads/")

pred <- predicted\_draws(mod, newdata = LH, ndraws = 1000)

########################## Results graphs #################################

ggplot(pred, aes(x = Year, y = Pelts)) +

stat\_lineribbon(aes(y = .prediction),

.width = c(.99, .95, .8, .5),

color = "#08519C") +

geom\_point(data = pred) + labs(y = "Pelts (thousands)") +

scale\_fill\_brewer() + facet\_wrap(~ Population)

########################## Results reports ################################

#...