# Script.R: R-Code to reproduce the results of

# Title: "Responses of earthworms to repeated exposure of three biocides applied singly and in a mixture in an agricultural field"

# Authors: Lisbeth Schnug, Torbjørn Ergon, Lena Jakob, Erik J. Joner,

# Hans Petter Leinaas

# Submitted to “*Science of the Total Environment*” (2014)

# The script was written by Torbjørn Ergon and Lisbeth Schnug

# Corresponding authors:

# Torbjørn Ergon

# Centre for Ecological and Evolutionary Synthesis,

# Department of Biosciences, University of Oslo, NORWAY

# t.h.ergon@ibv.uio.no

# Lisbeth Schnug

# Bioforsk - Norwegian Institute for Agricultural and Environmental Research

# lisbeth.schnug@bioforsk.no

# The code can be applied to the supplied data file "Input\_WORMS.csv"

# See README.txt for details

# The following R-packages must be installed for analyses:

# glmmADMB

# Hmisc

# Content of the script:

# 1. Load and transform data

# 2. Model selection

# 3. Full model

# 4. Model predictions (producing plots)

# 5. Responses at lowest concentration and Eisenia-EC50s

# 6. Calculation of slopes of regression lines

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

## 1. Load and transform data ##

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

# load package for generalized linear mixed models

library(glmmADMB)

library(Hmisc)

# load data

WORMS = read.csv("Input\_WORMS.csv")

# define the samples of each plot as factors

WORMS$Sample = factor(paste(WORMS$Sample, 1:length(WORMS$Sample)))

# Separate data samples from before (Pre) and after first biocide

# application

WORMS$Pre = ifelse(WORMS$Group == "Pre", 1, 0)

# Standardize concentrations (originalcons) for each treatment.

(Cons.mean = with(WORMS, tapply(originalcons, treatment, mean)))

(Cons.sd = with(WORMS, tapply(originalcons, treatment, sd)))

(logCons.mean = with(WORMS, tapply(log(originalcons), treatment, mean)))

(logCons.sd = with(WORMS, tapply(log(originalcons), treatment, sd)))

# Create variables used for fitting the final full model:

# First for non-transformed concentrations (originalcons).

# E=Esfenvalerate, P=Picoxystrobin, T=Triclosan, M=Mixture

WORMS$E.Cons.st = ifelse(WORMS$treatment == "E", (WORMS$originalcons - Cons.mean["E"])/Cons.sd["E"], 0)

WORMS$P.Cons.st = ifelse(WORMS$treatment == "P", (WORMS$originalcons - Cons.mean["P"])/Cons.sd["P"], 0)

WORMS$T.Cons.st = ifelse(WORMS$treatment == "T", (WORMS$originalcons - Cons.mean["T"])/Cons.sd["T"], 0)

WORMS$M.Cons.st = ifelse(WORMS$treatment == "M", (WORMS$originalcons - Cons.mean["M"])/Cons.sd["M"], 0)

# Then for log-transformed concentrations (log(originalcons))

WORMS$E.logCons.st = ifelse(WORMS$treatment == "E", (log(WORMS$originalcons) - logCons.mean["E"])/logCons.sd["E"], 0)

WORMS$P.logCons.st = ifelse(WORMS$treatment == "P", (log(WORMS$originalcons) - logCons.mean["P"])/logCons.sd["P"], 0)

WORMS$T.logCons.st = ifelse(WORMS$treatment == "T", (log(WORMS$originalcons) - logCons.mean["T"])/logCons.sd["T"], 0)

WORMS$M.logCons.st = ifelse(WORMS$treatment == "M", (log(WORMS$originalcons) - logCons.mean["M"])/logCons.sd["M"], 0)

# Collect the above variables in 2 new variables

WORMS$Cons.st = 0

WORMS$Cons.st[WORMS$treatment == "E"] = WORMS$E.Cons.st[WORMS$treatment == "E"]

WORMS$Cons.st[WORMS$treatment == "P"] = WORMS$P.Cons.st[WORMS$treatment == "P"]

WORMS$Cons.st[WORMS$treatment == "T"] = WORMS$T.Cons.st[WORMS$treatment == "T"]

WORMS$Cons.st[WORMS$treatment == "M"] = WORMS$M.Cons.st[WORMS$treatment == "M"]

WORMS$logCons.st = 0

WORMS$logCons.st[WORMS$treatment == "E"] = WORMS$E.logCons.st[WORMS$treatment == "E"]

WORMS$logCons.st[WORMS$treatment == "P"] = WORMS$P.logCons.st[WORMS$treatment == "P"]

WORMS$logCons.st[WORMS$treatment == "T"] = WORMS$T.logCons.st[WORMS$treatment == "T"]

WORMS$logCons.st[WORMS$treatment == "M"] = WORMS$M.logCons.st[WORMS$treatment == "M"]

# Create function for computing AICc

AICc = function(fit){

k = length(coef(fit))

n = fit$n

2\*k - 2\*fit$loglik + 2\*k\*(k+1)/(n-k-1)

}

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

## 2. Model selection ##

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

# To simplify model selection, data on each biocide was first fitted

# separately.

# For each biocide, three functional response types (A-C) were used (see

# below), and for each of these all models having either additive

# or interacting effects oy Year, Season and biocide concentration. This

# results in 9 models for each response type:

# A: Linear models with treatment intercepts forced through the control

models.forced = c(

"Pre + Year\*Season\*Cons.st + (1|Plot/Sample)",

"Pre + Year\*Season + Year\*Cons.st + Season\*Cons.st + (1|Plot/Sample)",

"Pre + Year\*Cons.st + Season\*Cons.st + (1|Plot/Sample)",

"Pre + Year\*Season + Season\*Cons.st + (1|Plot/Sample)",

"Pre + Year\*Season + Year\*Cons.st + (1|Plot/Sample)",

"Pre + Cons.st + Year\*Season + (1|Plot/Sample)",

"Pre + Season + Year\*Cons.st + (1|Plot/Sample)",

"Pre + Year + Season\*Cons.st + (1|Plot/Sample)",

"Pre + Year + Season + Cons.st + (1|Plot/Sample)")

# B: Linear models with free intercepts

models.free = c(

"Pre + treatment\*Year\*Season + Year\*Season\*Cons.st + (1|Plot/Sample)",

"Pre + treatment\*Year\*Season + Year\*Cons.st + Season\*Cons.st + (1|Plot/Sample)",

"Pre + treatment\*Year + treatment\*Season + Year\*Cons.st + Season\*Cons.st + (1|Plot/Sample)",

"Pre + treatment\*Year\*Season + Season\*Cons.st + (1|Plot/Sample)",

"Pre + treatment\*Year\*Season + Year\*Cons.st + (1|Plot/Sample)",

"Pre + Cons.st + treatment\*Year\*Season + (1|Plot/Sample)",

"Pre + treatment\*Season + Year\*Cons.st + (1|Plot/Sample)",

"Pre + treatment\*Year + Season\*Cons.st + (1|Plot/Sample)",

"Pre + treatment\*Year + treatment\*Season + Cons.st + (1|Plot/Sample)")

# C: Models with log-transformed concentrations and free intercepts

models.logcons = c(

"Pre + treatment\*Year\*Season + Year\*Season\*logCons.st + (1|Plot/Sample)",

"Pre + treatment\*Year\*Season + Year\*logCons.st + Season\*logCons.st + (1|Plot/Sample)",

"Pre + treatment\*Year + treatment\*Season + Year\*logCons.st + Season\*logCons.st + (1|Plot/Sample)",

"Pre + treatment\*Year\*Season + Season\*logCons.st + (1|Plot/Sample)",

"Pre + treatment\*Year\*Season + Year\*logCons.st + (1|Plot/Sample)",

"Pre + logCons.st + treatment\*Year\*Season + (1|Plot/Sample)",

"Pre + treatment\*Season + Year\*logCons.st + (1|Plot/Sample)",

"Pre + treatment\*Year + Season\*logCons.st + (1|Plot/Sample)",

"Pre + treatment\*Year + treatment\*Season + logCons.st + (1|Plot/Sample)")

# Building a function for running the above models

run.models = function(Data, agent, control.agent, Cons.st.controll, models){

D = Data[Data$treatment == agent | Data$treatment == control.agent,]

D$Cons.st[D$treatment == control.agent] = Cons.st.controll

# To save models, corresponding AICc values and covergence (TRUE/FALSE) create a data.frame

Models = data.frame(model = models, AICc = NA, stringsAsFactors=F, convergence=NA)

# Run models and save results in a list

fits = vector("list", nrow(Models))

for(i in 1:nrow(Models)){

cat(i)

fits[[i]] = try(glmmadmb(formula(paste("sum\_individuals ~ ", Models[i,"model"])), family = "nbinom", data=D))

# (When the proportion of juveniles is analyzed, a beta-binomial model

# must be chosen. The above line must then be changed to:

# fits[[i]] = try(glmmadmb(formula(paste("cbind(sum\_JUV, sum\_AD) ~ ",

# Models[i,"model"])), family = "binomial", data=D))

# with sum\_JUV being the juveniles in the sample and sum\_AD the adult

# individuals)

cat(".")

Models[i,"AICc"] = try(AICc(fits[[i]]))

cat(".")

Models[i,"convergence"] = try(fits[[i]]$conv==0)

}

return(list(agent=agent, Models=Models, fits=fits))

}

models = c(models.forced, models.free, models.logcons)

# Model selcection for Esfenvalerate (E)

E.models = run.models(Data = WORMS, agent="E", control.agent="1K", Cons.st.controll=-Cons.mean["E"]/Cons.sd["E"], models=models)

E.models$Models[order(E.models$Models$AICc),]

# Model selection for Picoxystrobin (P)

P.models = run.models(Data = WORMS, agent="P", control.agent="1K", Cons.st.controll=-Cons.mean["P"]/Cons.sd["P"], models=models)

P.models$Models[order(P.models$Models$AICc),]

# For triclosan and the mixture the control agent must be changed to

# Acetone (A)

WORMS2 = WORMS

WORMS2$treatment[WORMS2$Pre==1] = "A"

# Model selection for Triclosan (T)

T.models = run.models(Data = WORMS2, agent="T", control.agent="A", Cons.st.controll=-Cons.mean["T"]/Cons.sd["T"], models=models)

T.models$Models[order(T.models$Models$AICc),]

# Model selection for the Mixture (M)

M.models = run.models(Data = WORMS2, agent="M", control.agent="A", Cons.st.controll=-Cons.mean["M"]/Cons.sd["M"], models=models)

M.models$Models[order(M.models$Models$AICc),]

# Models with lowest AICc:

E.models$Models[E.models$Models$AICc==min(E.models$Models$AICc),] # 23 Pre + treatment\*Year\*Season + Year\*logCons.st + (1|Plot/Sample) 669.3205

P.models$Models[P.models$Models$AICc==min(P.models$Models$AICc),] # 14 Pre + treatment\*Year\*Season + Year\*Cons.st + (1|Plot/Sample) 639.6905

T.models$Models[T.models$Models$AICc==min(T.models$Models$AICc),] # 10 Pre + treatment\*Year\*Season + Year\*Season\*Cons.st + (1|Plot/Sample) 725.876888888889

M.models$Models[M.models$Models$AICc==min(M.models$Models$AICc),] # 24 Pre + logCons.st + treatment\*Year\*Season + (1|Plot/Sample) 608.2714

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

## 3. Full model ##

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

# The full model is constructed based on the single biocide models with

# the lowest AICc

fit.all = glmmadmb(sum\_individuals ~ treatment\*Year\*Season + Year\*(E.logCons.st + P.Cons.st) + Year\*Season\*T.Cons.st + M.logCons.st + Pre + (1|Plot/Sample), family = "nbinom", data=WORMS)

# Model ouput

summary(fit.all)

# Creating a plot for showing the degree of overdispersion of the data

var.nbin = fitted(fit.all)\*(1 + fitted(fit.all)/fit.all$alpha)

max(var.nbin)

plot(fitted(fit.all), var.nbin, col="red", ylab="Expected residual variance", xlab="Model prediction")

points(fitted(fit.all), fitted(fit.all), col="blue")

legend("topleft", c("Poison model", "Neg. bin. model"), pch=1, col = c("red", "blue"))

# (For beta-binomial models, overdispersion will depend on the number of

# trials, i.e. number if individuals (N):

# N = with(WORMS, seq(min(sum\_individuals), max(sum\_individuals), length.out=100))

# Relative overdispersion (ro):

# ro = (fit.full.JuvAd$alpha+N)/(fit.full.JuvAd$alpha+1)

# plot(N, ro)

# max(ro)

# )

# Variance between plots:

exp(2\*1.96\*sqrt(fit.all$S$Plot))

# Variance between samples within plots

exp(2\*1.96\*sqrt(fit.all$S$'Plot:Sample'))

# Assessing the goodness of fit of the full model by residual plots

# Residual plots

# For whole data set

plot(fitted(fit.all), residuals(fit.all, type="response"))

x = 1:40

# Comparing to the central 95% residual range in a Poisson distribution

lines(x, qpois(.975, x) - x)

lines(x, qpois(.025, x) - x)

# Calculate proportion of variance on link-scale in log(number of

# individuals) among plots explained by the fixed effects of the model

# Variance in mean fixed effects among plots

fixed.eff = log(fitted(fit.all))

var.fixed.eff.plot = var(tapply(fixed.eff, WORMS$Plot, mean))

# Residual random variance among plots

var.random = fit.all$S$Plot

# proportion of variance in log(number of individuals) among plots

# explained

var.fixed.eff.plot/(var.fixed.eff.plot + var.random)

# (To calculate the proportion of variance for beta-binomial models use:

# logit = function(p) log(p/(1-p))

# fixed.eff = logit(fitted(fit.full.JuvAd))

# var.fixed.eff.plot = var(tapply(fixed.eff, WORMS$Plot, mean))

# Variance in mean fixed effects among plots

# var.random = fit.full.JuvAd$S$Plot # Residual random variance among plots

# var.fixed.eff.plot/(var.fixed.eff.plot + var.random)

# )

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

## 4. Model predictions ##

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

### A. Predcition for control (K) and solvent (acetone) control (A)

# Calculate predictions for K and A

NewData.AK = expand.grid(Year = levels(WORMS$Year), Season = levels(WORMS$Season), treatment = factor(c("1K","A"), levels(WORMS$treatment)))

NewData.AK$E.logCons.st = 0

NewData.AK$P.Cons.st = 0

NewData.AK$T.Cons.st = 0

NewData.AK$M.logCons.st = 0

NewData.AK$Pre=0

pred.AK = predict(fit.all, NewData.AK, type = "response", interval = "confidence")

(Pred.AK = data.frame(NewData.AK, pred.AK))

# Plot predictions for spring 2010

library(Hmisc)

par(mfrow=c(2,2))

with(Pred.AK[Pred.AK$Year == "Y1" & Pred.AK$Season == "S",], errbar(c(1,2), fit, lwr, upr, xlim=c(0.5,2.5), ylim=c(0,45), axes=F, xlab="", ylab="# individuals"))

box()

axis(2)

axis(1, c(1,2), c("A", "K"))

title("Year 1 - Spring")

# Plot predictions for autumn 2010

with(Pred.AK[Pred.AK$Year == "Y1" & Pred.AK$Season == "A",], errbar(c(1,2), fit, lwr, upr, xlim=c(0.5,2.5), ylim=c(0,45), axes=F, xlab="", ylab="# individuals"))

box()

axis(2)

axis(1, c(1,2), c("A", "K"))

title("Year 1 - Autumn")

# Plot predictions for spring 2011

with(Pred.AK[Pred.AK$Year == "Y2" & Pred.AK$Season == "S",], errbar(c(1,2), fit, lwr, upr, xlim=c(0.5,2.5), ylim=c(0,45), axes=F, xlab="", ylab="# individuals"))

box()

axis(2)

axis(1, c(1,2), c("A", "K"))

title("Year 2 - Spring")

# Plot predictions for autumn 2011

with(Pred.AK[Pred.AK$Year == "Y2" & Pred.AK$Season == "A",], errbar(c(1,2), fit, lwr, upr, xlim=c(0.5,2.5), ylim=c(0,45), axes=F, xlab="", ylab="# individuals"))

box()

axis(2)

axis(1, c(1,2), c("A", "K"))

title("Year 2 - Autumn")

### B. Predcition for Esfenvalerate (E)

# Calculate predictions for E

x.E = with(WORMS[WORMS$treatment=="E",], seq(min(originalcons), max(originalcons), length.out=100))

NewData.E = expand.grid(E.cons = x.E, Year = levels(WORMS$Year), Season = levels(WORMS$Season))

NewData.E$treatment = factor("E", levels(WORMS$treatment))

NewData.E$E.logCons.st = (log(NewData.E$E.cons) - logCons.mean["E"])/logCons.sd["E"]

NewData.E$P.Cons.st = 0

NewData.E$T.Cons.st = 0

NewData.E$M.logCons.st = 0

NewData.E$Pre=0

pred.E = predict(fit.all, NewData.E, type = "response", interval = "confidence")

Pred.E = data.frame(NewData.E, pred.E)

# Plot predictions for spring 2010

windows();

par(mfrow=c(2,2), oma=c(.1,.1,.1,.1), mar=c(4,4,2,2))

with(Pred.E[Pred.E$Year == "Y1" & Pred.E$Season == "S",], plot(E.cons, fit, type="l", ylim=c(0,40), ylab="Individuals/0.25m2", xlab=""))

with(Pred.E[Pred.E$Year == "Y1" & Pred.E$Season == "S",], lines(E.cons, lwr, lty=2))

with(Pred.E[Pred.E$Year == "Y1" & Pred.E$Season == "S",], lines(E.cons, upr, lty=2))

# Insert estimates for K som reference

abline(h=Pred.AK[Pred.AK$Year == "Y1" & Pred.AK$Season == "S" & Pred.AK$treatment == "1K", "fit"], col="red")

abline(h=Pred.AK[Pred.AK$Year == "Y1" & Pred.AK$Season == "S" & Pred.AK$treatment == "1K", "lwr"], col="red", lty=2)

abline(h=Pred.AK[Pred.AK$Year == "Y1" & Pred.AK$Season == "S" & Pred.AK$treatment == "1K", "upr"], col="red", lty=2)

title("Spring 2010")

# Plot predictions for autumn 2010

with(Pred.E[Pred.E$Year == "Y1" & Pred.E$Season == "A",], plot(E.cons, fit, type="l", ylim=c(0,40), ylab="", xlab=""))

with(Pred.E[Pred.E$Year == "Y1" & Pred.E$Season == "A",], lines(E.cons, lwr, lty=2))

with(Pred.E[Pred.E$Year == "Y1" & Pred.E$Season == "A",], lines(E.cons, upr, lty=2))

# Insert estimates for K som reference

abline(h=Pred.AK[Pred.AK$Year == "Y1" & Pred.AK$Season == "A" & Pred.AK$treatment == "1K", "fit"], col="red")

abline(h=Pred.AK[Pred.AK$Year == "Y1" & Pred.AK$Season == "A" & Pred.AK$treatment == "1K", "lwr"], col="red", lty=2)

abline(h=Pred.AK[Pred.AK$Year == "Y1" & Pred.AK$Season == "A" & Pred.AK$treatment == "1K", "upr"], col="red", lty=2)

title("Autumn 2010")

# Plot predictions for spring 2011

with(Pred.E[Pred.E$Year == "Y2" & Pred.E$Season == "S",], plot(E.cons, fit, type="l", ylim=c(0,45), ylab="Individuals/0.25m2", xlab="Esfenvalerate (µmol/kg)" ))

with(Pred.E[Pred.E$Year == "Y2" & Pred.E$Season == "S",], lines(E.cons, lwr, lty=2))

with(Pred.E[Pred.E$Year == "Y2" & Pred.E$Season == "S",], lines(E.cons, upr, lty=2))

# Insert estimates for K som reference

abline(h=Pred.AK[Pred.AK$Year == "Y2" & Pred.AK$Season == "S" & Pred.AK$treatment == "1K", "fit"], col="red")

abline(h=Pred.AK[Pred.AK$Year == "Y2" & Pred.AK$Season == "S" & Pred.AK$treatment == "1K", "lwr"], col="red", lty=2)

abline(h=Pred.AK[Pred.AK$Year == "Y2" & Pred.AK$Season == "S" & Pred.AK$treatment == "1K", "upr"], col="red", lty=2)

title("Spring 2011")

# Plot predictions for autmn 2011

with(Pred.E[Pred.E$Year == "Y2" & Pred.E$Season == "A",], plot(E.cons, fit, type="l", ylim=c(0,45), xlab="Esfenvalerate (µmol/kg)", ylab=""))

with(Pred.E[Pred.E$Year == "Y2" & Pred.E$Season == "A",], lines(E.cons, lwr, lty=2))

with(Pred.E[Pred.E$Year == "Y2" & Pred.E$Season == "A",], lines(E.cons, upr, lty=2))

# Insert estimates for K som reference

abline(h=Pred.AK[Pred.AK$Year == "Y2" & Pred.AK$Season == "A" & Pred.AK$treatment == "1K", "fit"], col="red")

abline(h=Pred.AK[Pred.AK$Year == "Y2" & Pred.AK$Season == "A" & Pred.AK$treatment == "1K", "lwr"], col="red", lty=2)

abline(h=Pred.AK[Pred.AK$Year == "Y2" & Pred.AK$Season == "A" & Pred.AK$treatment == "1K", "upr"], col="red", lty=2)

title("Autumn 2011")

mtext("Treatment E", outer=T, line=-1, cex=2)

# Prediction for the other treatments are calculated and plotted

# accordingly.

# NB! In the case of triclosan and the mixture the control agent must be

# changed to A

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

## 5. Responses at lowest concentration and Eisenia-EC50s ##

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

# Responses are calculated as relative change in log(number of

# individuals)

# Extract lowest concentration of all seasons and years

with(WORMS, tapply(originalcons, treatment, range))

lowest = with(WORMS, tapply(originalcons, treatment, min))

lowest.st = (lowest - Cons.mean)/Cons.sd

# log-transform lowest concentration

loglowest = with(WORMS, tapply(log(originalcons), treatment, min))

loglowest.st = (loglowest - logCons.mean)/logCons.sd

### Esfenvalerate Spring 2010

# A: calculate response at lowest concentration

# Create a data.framen with 2 lines. The first line is for the tretament

# and the second for the control

D = data.frame(

treatment = factor(c("E","1K"), levels=levels(WORMS$treatment)),

Year = factor(rep("Y1",2), levels=levels(WORMS$Year)),

Season = factor(rep("S",2), levels=levels(WORMS$Season)),

P.Cons.st = c(0,0),

T.Cons.st = c(0,0),

M.logCons.st = c(0,0),

E.logCons.st = c(loglowest.st["E"],0),

Pre = c(0,0))

D

# The respective full model is inserted in the model matrix

X = model.matrix(~ treatment\*Year\*Season + Year\*(E.logCons.st + P.Cons.st) + Year\*Season\*T.Cons.st + M.logCons.st + Pre, D)

X.diff = matrix(X[1,] - X[2,], 1)

diff = X.diff %\*% coef(fit.all)

se.diff = sqrt(diag(X.diff %\*% vcov(fit.all) %\*% t(X.diff)))

# calculate the contrast and corresponding 95% confidence intervals

exp(diff + c(0,-1,1)\*1.96\*se.diff)

# Express contrast as relative change

exp(diff + c(0,-1,1)\*1.96\*se.diff) - 1

# B: Calculate response at the Eisenia-EC50-concentration

D = data.frame(

treatment = factor(c("E","1K"), levels=levels(WORMS$treatment)),

Year = factor(rep("Y1",2), levels=levels(WORMS$Year)),

Season = factor(rep("S",2), levels=levels(WORMS$Season)),

P.Cons.st = c(0,0),

T.Cons.st = c(0,0),

M.logCons.st = c(0,0),

E.logCons.st = c((log(33.5) - logCons.mean["E"])/logCons.sd["E"],0), # # The respective EC50 value is used instead of the lowest concentration

Pre = c(0,0))

D

X = model.matrix(~ treatment\*Year\*Season + Year\*(E.logCons.st + P.Cons.st) + Year\*Season\*T.Cons.st + M.logCons.st + Pre, D)

X.diff = matrix(X[1,] - X[2,], 1)

diff = X.diff %\*% coef(fit.all)

se.diff = sqrt(diag(X.diff %\*% vcov(fit.all) %\*% t(X.diff)))

# calculate the contrast and corresponding 95% confidence intervals

exp(diff + c(0,-1,1)\*1.96\*se.diff)

# Express this as relative change

exp(diff + c(0,-1,1)\*1.96\*se.diff) - 1

# The responses at concentrations of the other biocides are calculated

# accordingly

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

## 6. Calculation of slopes of regression lines ##

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

### Slopes for Picoxystrobin in 2010 (Seasons are the same)

coef(fit.all)

x = c( # 1 for the parameters to be included in the slope, 0 for the remaining

0, #(Intercept) 2.2188 0.1314 16.88 < 2e-16 \*\*\*

0, # treatmentA 0.1143 0.1798 0.64 0.52503

0, # treatmentE -1.3899 0.2524 -5.51 3.6e-08 \*\*\*

0, # treatmentM -1.7854 0.2952 -6.05 1.5e-09 \*\*\*

0, # treatmentP -0.8845 0.2416 -3.66 0.00025 \*\*\*

0, # treatmentT 0.1165 0.1878 0.62 0.53489

0, # YearY2 0.1103 0.1478 0.75 0.45537

0, # SeasonS -0.8976 0.1963 -4.57 4.8e-06 \*\*\*

0, # E.logCons.st -0.0328 0.1446 -0.23 0.82047

1, # P.Cons.st -0.1916 0.1722 -1.11 0.26592

0, # T.Cons.st 0.1993 0.0952 2.09 0.03622 \*

0, # M.logCons.st -0.4981 0.1695 -2.94 0.00330 \*\*

0, # Pre 0.3985 0.1860 2.14 0.03218 \*

0, # treatmentA:YearY2 -0.3160 0.2097 -1.51 0.13185

0, # treatmentE:YearY2 -0.4398 0.3463 -1.27 0.20406

0, # treatmentM:YearY2 -1.0952 0.5011 -2.19 0.02885 \*

0, # treatmentP:YearY2 -5.7698 1.1343 -5.09 3.6e-07 \*\*\*

0, # treatmentT:YearY2 -0.9719 0.3656 -2.66 0.00786 \*\*

0, # treatmentA:SeasonS -0.5647 0.2989 -1.89 0.05886 .

0, # treatmentE:SeasonS 0.9755 0.3422 2.85 0.00436 \*\*

0, # treatmentM:SeasonS 0.5912 0.4056 1.46 0.14493

0, # treatmentP:SeasonS 0.1717 0.3646 0.47 0.63781

0, # treatmentT:SeasonS -0.3494 0.2998 -1.17 0.24375

0, # YearY2:SeasonS 1.6388 0.2329 7.04 2.0e-12 \*\*\*

0, # YearY2:E.logCons.st -0.4573 0.2322 -1.97 0.04892 \*

0, # YearY2:P.Cons.st -4.7148 0.9830 -4.80 1.6e-06 \*\*\*

0, # YearY2:T.Cons.st -1.6852 0.5187 -3.25 0.00116 \*\*

0, # SeasonS:T.Cons.st -0.1776 0.1712 -1.04 0.29980

0, # treatmentA:YearY2:SeasonS 0.4038 0.3525 1.15 0.25201

0, # treatmentE:YearY2:SeasonS -0.8229 0.4612 -1.78 0.07436 .

0, # treatmentM:YearY2:SeasonS -0.0313 0.6276 -0.05 0.96025

0, # treatmentP:YearY2:SeasonS -0.1448 0.5240 -0.28 0.78236

0, # treatmentT:YearY2:SeasonS 1.1650 0.4541 2.57 0.01031 \*

0) # YearY2:SeasonS:T.Cons.st 1.4460 0.5685 2.54 0.01097 \*

x = matrix(x,1)

# extract slope (NB! This si the slope per sd!)

slope.st = x %\*% coef(fit.all)

# Calculate the slope per 10 concentration units (µmol/kg)

slope10 = 10\*slope.st/Cons.sd["P"]

# Calculate the standard error of the slope

slope.st.se = sqrt(diag(x %\*% vcov(fit.all) %\*% t(x)))

slope10.se = 10\*slope.st.se/Cons.sd["P"]

# Find 95% confidence interval for the slope expressed as relative change per 10 concentration units

exp(slope10 + c(0,-1,1)\*1.96\*slope10.se)-1

# The slopes for the other treatments are calculated accordingly

# (For log-concentrations (as in the case of Esfenvalerate) the change in slope may be expressed per 10% increase in concentration

# Slope and SE:

# slope.logE = slope.st/logCons.sd["E"]

# slope.logE.se = slope.st.se/logCons.sd["E"]

# Calculate the resulting relative increase in response (and 95%

# confidence interval) when the concentration increases with 10%:

# 1.1^(slope.logE + c(0,-1,1)\*1.96\*slope.logE.se)-1

# )