```r
##############################################################################
## Title: Performance analysis, comparing model estimates to control data   ##
##                                                                          ##
## Script author: Kali Z. Mattingly                                         ##
##                                                                          ##
## Date created: 08-31-2022                                                 ##
##                                                                          ##
## Email: mattingly.kali@epa.gov                                            ##
##                                                                          ##
## Citation:  Haselman, J.T., Nichols, J.W., Mattingly, K.Z., Hornung, M.W. ##
##            and Degitz, S.J. 2023. A Biologically Based Computational     ##
##            Model for the Hypothalamic-Pituitary-Thyroid (HPT) Axis in    ##
##            Xenopus laevis larvae. Mathematical Biosciences (submitted)   ##
##                                                                          ##
##############################################################################

##### Before running these analyses, run "Scripts/Amphibian_Thyroid_Model.R"
# needed to load packages, `realdata`, `dynout` into environment #####

install.packages("reshape2")
install.packages("FME")
install.packages("tidyverse")

library(reshape2)
library(FME)
library(tidyverse)

##### Model performance analysis with FME::modCost() #####

# Make control data colnames match dynout
controldata <- realdata
colnames(controldata)[3] <- "time"
colnames(controldata)[4] <- "BW"
colnames(controldata)[5] <- "TG.MIT.pmoles"
colnames(controldata)[6] <- "TG.DIT.pmoles"
colnames(controldata)[7] <- "TG.T4.pmoles"
colnames(controldata)[8] <- "CT3TOTPL"
colnames(controldata)[9] <- "CT4TOTPL"
colnames(controldata)[10] <- "FCNTOT"

# Convert data table into long form (i.e., all measurements in a single column with
# new column indicating previous column heading)
long.controldata <- melt(data.table(controldata), id.vars = c(1,3), measure.vars=c(4:10))


# Remove NAs
na_vec <- which(!complete.cases(long.controldata))
long.controldata_no_na <- long.controldata[-na_vec,]

# Consolidate by time point, across chemicals
all.controldata <- long.controldata_no_na %>%
  group_by(variable, time) %>%
  summarise(
    mean = mean(value, na.rm = TRUE),
    sd = sd(value, na.rm = TRUE),
    n=n()
  )
all.controldata <- as.data.frame(all.controldata)

# Replace sd=0 with 1, because sd=0 creates a zero-in-denominator problem in modCost,
# where res = (mod - obs)/sd
all.controldata$sd[all.controldata$sd==0] <- 1

# Get the first variable to initialize the cost
sub.controldata <- all.controldata[all.controldata$variable=="BW", ]
sub.controldata <- sub.controldata[,c(2:4)]
```

```r
colnames(sub.controldata)[2] <- "BW"
cost <- modCost(model=dynout, obs=sub.controldata, err="sd")

# Cycle through remaining variables
for (i in c(2:length(levels(all.controldata$variable)))) {

  # subset data to variable i
  var <- levels(all.controldata$variable)[i]
  sub.controldata <- all.controldata[all.controldata$variable==var, ]
  sub.controldata <- sub.controldata[,c(2:4)]
  colnames(sub.controldata)[2] <- var

  # redo cost, adding each variable
  cost <- modCost(model=dynout, obs=sub.controldata, err="sd", cost=cost)
}

##### Results #####

# total model cost
cost$model

# cost of each variable = sum of squared weighted residuals for a variable
variableSSR <- cost$var
variableSSR$SSRTot <- cost$model
variableSSR$SSRratio <- variableSSR$SSR/variableSSR$SSRTot
variableSSR

SSR_plot <- ggplot(variableSSR, aes(x=name, y=SSRratio))
+ geom_point(aes(col=name, cex=3)) + xlab("")
+ theme(legend.position="none", axis.text.x=element_text(angle=45, hjust=1))
+ geom_label(aes(label=round(SSR, 3)), nudge_y=0.1, size=3)
print(SSR_plot)

# residual = (mod - obs)/sd
allresid <- cost$residuals

mean(allresid$res)

allresid %>%
  group_by(name) %>%
  summarise(
    group_mean_residual = mean(res, na.rm = TRUE)
  )

allresid %>%
  group_by(x) %>%
  summarise(
    group_mean_residual = mean(res, na.rm = TRUE)
  )

Residual_plot <- ggplot(allresid, aes(x=x, y=res))
+ geom_point(aes(col=name))
+ geom_line(aes(group=name, col=name))
+ ylab("Weighted residual = (mod - obs)/sd")
+ xlab("Time (hr)")
print(Residual_plot)

```