ICE PBPK workflow

Description

The workflow allows the flexibility to select from three different rat and/or human PK models from the EPA's httk package: solve_pbtk, solve_gas_pbtk, and solve_fetal_pbtk. The workflow is to predict plasma and tissue concentration profiles across time following a dose of a substance. Exposure routes include oral ingestion (solve_pbtk, solve_fetal_pbtk), intrevenous (solve_pbtk, solve_fetal_pbtk), and inhalation (solve_gas_pbtk). For inhalation exposures using the dose is provided as an air concentration, uM or ppmv.

An example input file is included: ChemicalData_Rnotebook.txt

Load libraries

```r
#load libraries
library(tidyverse)
library(rlang)
library(deSolve)
library(doParallel)
library(httk) #this is needed for models: solve_pbtk, solve_gas_pbtk, solve_fetal_pbtk.
              #The code is compatible with httk_2.0.2
library(openxlsx) #for writing the excel file that is user output
library(data.table)
```

Define custom functions

```r
#function to parse time series list into second output table format.
#The input for this function is a list object (x) that contains an
#entry with the pbpk outputs from the httk functions along with the
#plasma CSS and CASRN for the chemical

## assemble.table_* function is to assemble direct outputs from different models to a format that is us
## Because the number of output columns are different for each model, so the assemble.table_* is modifi
#nOutParam - total number of time column and predicted tissue concentration column in the output table
#nStartCol - start column number, including time column
#nStartConCol - start column number of tissue concentration , excluding time column
#nTotalCol - number of total columns in the output table
#assemble.table <- function(x, nOutParam=10, nStartCol=4, nStartConCol=5, nTotalCol=13){
assemble.table_pbtk <- function(x, nOutParam=NULL, nStartCol=NULL, nStartConCol=NULL, nTotalCol=NULL){
  CASRN <- rep(x$CASRN[1], nOutParam)                               #x is "output" data frame; 9 rows, 8
  compound <- rep(chem.physical_and_invitro.data[which(
    chem.physical_and_invitro.data$CAS==x$CASRN[1]),1], nOutParam) #extract the "compound" information
  dtxsid <- rep(chem.physical_and_invitro.data[which(
    chem.physical_and_invitro.data$CAS==x$CASRN[1]),4], nOutParam) #extract the "DTXSID" information fo
  half_life_h <- rep(x$half_life_h[1], nOutParam)
  maxAUC.h <- rep(x$maxAUC.h[1], nOutParam)
  CSS_plasma <- rep(x$CSS_plasma[1], nOutParam)
  Cmax <- c(NA, sapply(as.data.frame(x[nStartConCol:nTotalCol]), function(y) max(as.numeric(y)))) #from
  Unit <- rep(ConcentrationUnit, nOutParam)
  compartment <- names(x)[nStartCol:nTotalCol]                           #nStartCol - start column number, in
  concs <- t(x[nStartCol:nTotalCol])                                     #nTotalCol - number of total columns

  out <- as.data.frame(cbind(CASRN, compound, dtxsid, half_life_h, maxAUC.h, CSS_plasma, Cmax, Unit, com
  rownames(out) <- NULL
  return(out)
}

assemble.table_gas <- function(x, nOutParam=NULL, nStartCol=NULL, nStartConCol=NULL, nTotalCol=NULL){
  CASRN <- rep(x$CASRN[1], nOutParam)                                   #x is "output" data frame; 9 rows, 8
  compound <- rep(chem.physical_and_invitro.data[which(
    chem.physical_and_invitro.data$CAS==x$CASRN[1]),1], nOutParam) #extract the "compound" information f
  dtxsid <- rep(chem.physical_and_invitro.data[which(
    chem.physical_and_invitro.data$CAS==x$CASRN[1]),4], nOutParam) #extract the "DTXSID" information fo
  half_life_h <- rep(x$half_life_h[1], nOutParam)
  maxAUC.h <- rep(x$maxAUC.h[1], nOutParam)
  #CSS_plasma <- rep(x$CSS_plasma[1], nOutParam)
  Cmax <- c(NA, sapply(as.data.frame(x[nStartConCol:nTotalCol]), function(y) max(as.numeric(y)))) #from
  Unit <- rep(ConcentrationUnit, nOutParam)
  compartment <- names(x)[nStartCol:nTotalCol]                           #nStartCol - start column number, in
  concs <- t(x[nStartCol:nTotalCol])                                     #nTotalCol - number of total columns

  out <- as.data.frame(cbind(CASRN, compound, dtxsid, half_life_h, maxAUC.h, Cmax, Unit, compartment, co
  rownames(out) <- NULL
  return(out)
}

assemble.table_fetal <- function(x, nOutParam=NULL, nStartCol=NULL, nStartConCol=NULL, nTotalCol=NULL){
  CASRN <- rep(x$CASRN[1], nOutParam)                                    #x is "output" data frame; 21 rows,
  compound <- rep(chem.physical_and_invitro.data[which(
    chem.physical_and_invitro.data$CAS==x$CASRN[1]),1], nOutParam) #extract the "compound" information f
  dtxsid <- rep(chem.physical_and_invitro.data[which(
    chem.physical_and_invitro.data$CAS==x$CASRN[1]),4], nOutParam) #extract the "DTXSID" information fo
  half_life_h <- rep(x$half_life_h[1], nOutParam)
  maxAUC.h <- rep(x$maxAUC.h[1], nOutParam)
  maxFetalAUC.h <- rep(x$maxfAUC.h[1], nOutParam)
  Cmax <- c(NA, sapply(as.data.frame(x[nStartConCol:nTotalCol]), function(y) max(as.numeric(y)))) #from
  Unit <- rep(ConcentrationUnit, nOutParam)
  compartment <- names(x)[nStartCol:nTotalCol]                           #nStartCol - start column number, in
  concs <- t(x[nStartCol:nTotalCol])                                     #nTotalCol - number of total columns

  out <- as.data.frame(cbind(CASRN, compound, dtxsid, half_life_h, maxAUC.h, maxFetalAUC.h, Cmax, Unit,
  rownames(out) <- NULL
  return(out)
}
```

Input data and output file path

There are several input variables needed to run the code. Some variables are model specific, as detailed. Adjust the file paths to point to the file in your directory.

chemFile: The file containing chemical data from ICE. this has the CASRN as the first field

```
chemFile <- "ChemicalData_Rnotebook.txt"   # chemicals data from ICE, include CASRN field as identifier
Userout <- "User_Results.xlsx"                    # file name (and path) for the output
                                                  # file for the tissue concentrations
output_image <-"User_plot.pdf"                    # the file for the plot of the results
```

Model variables

Details about what model, route, and dose need to be specified.

What model?

Currently, species in ICE is limited to human or rat. Using this notebook one can expand the species with minor editing of the code, provided the parameters are available.

```
species <- "human"                                # "human" or "rat", the solve_fetal_pbtk model is onl
modelType <-  "solve_fetal_pbtk"                    # "solve_pbtk", "solve_fetal_pbtk", or "solve_gas_p
```

For the PBPK models some additional parameters can be modified. An inhalation exposure has additional parameters. The route determines where the chemical will enter the system. Exposure route, interval, and days are needed for all PBPK models.

```
route <- "iv"      # options are "inhalation", "iv","oral"; "solve_gas_pbtk" uses the inhalation (gas)
bWeight <- 70 # unit is kg;the body weight of simulated subject, default is 70 for human, and 0.25 for
interv <- 24  # dosing interval, hours
ndays <- 3    # number of days dosing is done
expDose <- 1  # current calculations assume 1mg/kg/dose for solve_pbtk, solve_fetal_pbtk model
expConc <- 1
ConcentrationUnit <- "uM" # options are uM and mg/L
```

Inhalation specific parameters

For an inhalation route of exposure, the “expDose” exposure route is currently not available. Instead, the “expConc” exposure route, which is more typical of a gas exposure, can be used. This models an inhaled concentration over a duration of time, set by “expLength”.

Note that “expDose” is currently not supported.

```
gasDosing <-"expConc" # "expConc" or "expDose"
#This is only needed if gasDosing =="expConc" bc otherwise default to expDose
if(gasDosing =="expConc" & (!exists("expConc") || is.null(expConc))){
  expConc <- 1 #current calcualtions assume 1mg/kg/dose
}
gasInputUnit <- "uM" # Input concentration unit for gas model (httkv2.1.0 and later), "uM" or "ppmv" (d
expLength <- 0.25 #length of the gas exposure, hours
```

Gestation model (solve_fetal_pbtk) specific parameters

```
gestationDays <- 91 #gestational days when dose starts, range is 91 days (14 weeks) -  280 days (40 wee
```

Load data

Notice that the chemical input has the source of the FU and Clint data. ICE now has 2 different sources of these values so the source is important for tracking this information.

```
chemical <- as.data.frame(read_delim(chemFile, delim = "\t")) #there are single quotes
                      #in chemical names that need to be addressed, tidyR handles this well
chemical[1:2,]        #to check the format of input data
```

```
##   Substance Name      CASRN        DTXSID Chemical Parameters fu fu Source
## 1 Beclomethasone  4419-39-0 DTXSID5040750                  0.130     OPERA
## 2     Ketanserin 74050-98-9 DTXSID3023188                  0.052     OPERA
##   Chemical Parameters Clint ul/min/10^6 cells Clint Source
## 1                                    34.43499        OPERA
## 2                                    46.88134        OPERA
##   Chemical Parameters pKa, Acidic   Chemical Parameters LogP log10
## 1                             8.689                          1.980
## 2                             8.597                          3.282
##   Chemical Parameters MW g/mol  Chemical Parameters HL log10, atm-m3/mole
## 1                         408.916                                 -10.130
## 2                         395.427                                  -7.616
##   Chemical Parameters pKa, Basic
## 1                            6.546
## 2                            5.826
```

Preparing data

Minor prep work is done on the data that comes from ICE. Partitioning coefficients are obtained by internal function from the httk package using the provided phys chem parameters. Compare with the example input file for naming. Note that a few checks are done - these will generate flags that can be used to follow up as needed.

```
##column names should be labeled correctly. RegEx have been used to help, see the example file

#cleaning the names for taking from an ICE Query-stricter matching is used to handel including the ADME
colnames(chemical)<-gsub(".*Name.*", "ChemicalName", colnames(chemical))
colnames(chemical)<-gsub(".*Parameters fu.*", "fu", colnames(chemical), ignore.case =TRUE)
colnames(chemical)<-gsub(".*funbound.*", "fu", colnames(chemical), ignore.case =TRUE)
colnames(chemical)<-gsub(".*fu Source.*", "fu Source", colnames(chemical), ignore.case =TRUE)
colnames(chemical)<-gsub(".*MW.*", "MW", colnames(chemical))
colnames(chemical)<-gsub(".*HL.*", "HL", colnames(chemical))
colnames(chemical)<-gsub(".*Parameters Clint.*", "Clint", colnames(chemical), ignore.case = TRUE)
colnames(chemical)<-gsub(".*Clint Source.*", "Clint Source", colnames(chemical), ignore.case = TRUE)
colnames(chemical)<-gsub(".*LogP.*", "LogP", colnames(chemical))
colnames(chemical)<-gsub(".*pKa, Basic.*", "pka_Accept", colnames(chemical))
colnames(chemical)<-gsub(".*pKa, Acidic.*", "pka_Donor", colnames(chemical))

#Converting units for internsic clearance values
#chemical$Clint<-10^chemical$Clint #this moves from log10 ul/ml/10^6 cells
                                   #to just ul/ml/10^6 cells
chemical$logPwa<--1*chemical$HL    #this gets the water octanal coeff
#Add the flag here as a warning if model not appropriate or other issues related to generating predicti
chemical$Flag <-""
#catch for low Fu
chemical$Flag[chemical$fu<=1e-10]<-"Fu is zero,likely due to rounding. Setting to 1e-5"
chemical$fu[chemical$fu<=1e-10]<-1e-5

#setting up additional defaults:
if(!exists("expDose") || is.null(expDose)){
  expDose <- 1                    #current calculations assume 1mg/kg/dose
}
```

PBPK models

Currently, the PBPK analysis in ICE supports the “solve_pbtk”, “solve_gas_pbpk” and “solve_fetal_pbtk” model types. This is wrapper code to format the inputs and parameters needed to run the models specified by the httk package.

Parameter processing for the PBPK models

The httk package needs some additional parameters including the correct capitalization of the species and ensuring that the adding of the user-specified data to the chem table is properly integrated.

```
#library(httk)
##preprocessing variables:
if (tolower(species) == "rat") {
  species_1 <- "Rat"
}
if (tolower(species) == "human") {
  species_1 <- "Human"
}
options(stringsAsFactors = FALSE)
#add chemical info to the table. Using variable coming from ICE
#to deal with mapping issues
chemical$DTXSID2<-paste0(chemical$DTXSID, "_n")
if(species_1=="Human"){
  chem.physical_and_invitro.data <- add_chemtable(chemical, current.table = chem.physical_and_invitro.da
}else{
  chem.physical_and_invitro.data <- add_chemtable(chemical, current.table = chem.physical_and_invitro.da
  chem.physical_and_invitro.data <- add_chemtable(chemical, current.table = chem.physical_and_invitro.da
}

if(route != "iv"){
  iv.dose = FALSE
}else{iv.dose =TRUE}
```

```
dpd<-24/interv #calculating the doses per day
```

to allow the user to change the bodyweight for rat and human

```
if(species_1=="Human"){
  physiology.data[4,6] <- bWeight
}else{
  physiology.data[4,4] <- bWeight #the rat body weight
}
```

solve_pbtk model

The solve_pbtk model is a PBPK model that has gut lumen, gut, liver, kidneys, veins, arteries, lungs, and the rest of the body compartments.

```
if(modelType == "solve_pbtk"){
  # note that dose=0 stops the initial dosing of the compartments and is needed to accurately model the
  output_list <- list()
  count=0
  for(this.cas in chemical[,"CASRN"]) {
    half_life <- calc_half_life(chem.cas = this.cas, species = species_1)
    param <- parameterize_pbtk(chem.cas = this.cas, species = species_1, default.to.human = TRUE)
    print(param$BW)  # to make sure that the body weight had been changed.

    output0 <- solve_pbtk(chem.cas = this.cas, parameters = NULL, doses.per.day = dpd, days = ndays, ts
    output0 <- as.data.frame(output0)
    maxAUC.h <- max(output0$AUC)*24
    concs <- output0 %>% select(c("time", "Cgut", "Cliver", "Cven", "Clung", "Cart", "Crest", "Ckidney"

    if(route =="iv"){
      #ignore computational errors associated with the solver
        to_ignore <- 2 #<- time point 2 will always be anomalous
        n_to_ignore <- max(1,floor((24*ndays)/interv))
        if((24/interv) == 1){
          steps_between_doses <- 97
        }else if(24/interv > 1){
          steps_between_doses <- floor(97 / (24/interv))+1
        }else if(24/interv < 1){
          steps_between_doses <- floor(97 / (24/interv))
        }
        for(i in 1:n_to_ignore){
          to_ignore[i+1] = to_ignore[i] + steps_between_doses
        }
        concs <- concs[-to_ignore,]
    }

    #steady state concentration
    CSS0 <- calc_analytic_css(chem.cas = this.cas, parameters = NULL, daily.dose = expDose*dpd, doses.p
    #convert units and store output
    output_time.h <- sapply(concs[,1], function(x) x*24) #days to hours
    output_vals <- concs %>% select(c("Cgut", "Cliver", "Cven", "Clung", "Cart", "Crest", "Ckidney", "C
    output <- as.data.frame(cbind('CASRN'=rep(this.cas, length(output_time.h)),
                                  'half_life_h' = rep(half_life, length(output_time.h)),
                                  'maxAUC.h' = rep(maxAUC.h, length(output_time.h)),
                                  'CSS_plasma'=rep(CSS0, length(output_time.h)),
                                  output_time.h,
                                  output_vals))

     count <- count+1
     output_list[[count]] <- as.data.frame(output)

  }
}
```

solve_gas_pbtk model

The solve_gas_pbtk is a PBPK model similar to the "solve_pbtk" model but is uses an inhalation route of exposure assuming the chemical is volatile (gas). As a result, it has some addtional checks to see if the assumption of gas exposure is reasonable but will still proceed with a flag warning.

```
if(modelType == "solve_gas_pbtk"){
  #check the assumption of volatility
  chemical$Flag[chemical$HL <= -7.80388 & chemical$Flag!=""]<-"Fu is zero,likely due to rounding. Setti
            Chemical likely nonvolatile, consider appropriateness of model"
  chemical$Flag[chemical$HL <= -7.80388 & chemical$Flag==""]<-"Chemical likely nonvolatile, consider ap

  # note that dose=0 stops the initial dosing of the compartments and is needed to accurately model the
  output_list <- list()
  count=0
  for(this.cas in chemical[,"CASRN"]) {

      # --- expDose option has currently been dissabled due to a bug in the httk package ---
      # if(gasDosing == "expDose"){
      #   ConcentrationUnit <- "uM"  # output unit only has one option of "uM" for solve_gas_pbtk model
      #  concMax <- max(solve_gas_pbtk(chem.cas = this.cas, parameters = NULL, doses.per.day = dpd, day
      # }
    half_life <- calc_half_life(chem.cas = this.cas, species = species_1)
    if(gasDosing == "expConc"){
      output0 <- solve_gas_pbtk(chem.cas = this.cas, parameters = NULL, doses.per.day = NULL, days = nd
      output0 <- as.data.frame(output0)
      maxAUC.h <- max(output0$AUC)*24
      concs <- output0 %>% select(c("time", "Cgut", "Cliver", "Cven", "Clung", "Cart", "Crest", "Ckidne
    }
    #steady state concentration
    #CSS0 <- calc_analytic_css(chem.cas = this.cas, parameters = NULL, daily.dose = expDose*dpd, doses.

    #convert units and store output
    output_time.h <- sapply(concs[,1], function(x) x*24) #days to hours
    output_vals <- concs %>% select(c("Cgut", "Cliver", "Cven", "Clung", "Cart", "Crest", "Ckidney", "C
    output <- as.data.frame(cbind('CASRN'=rep(this.cas, length(output_time.h)),
                                  'half_life_h' = rep(half_life, length(output_time.h)),
                                  'maxAUC.h' = rep(maxAUC.h, length(output_time.h)),
                                  #'CSS_plasma'=rep(CSS0, length(output_time.h)),
                                  output_time.h,
                                  output_vals))

     count <- count+1
     output_list[[count]] <- as.data.frame(output)

  }
}
```

solve_fetal_pbtk model

The solve_fetal_pbtk is a human PBPK model that includes both maternal and fetal compartments and a placenta modeled as a joint organ shared by mother and fetus. It simulated gestation day of 91-280 days. A flag warning will show if the gestation day is out of this range. The exposure routes for this model are oral and IV.

```
if(modelType == "solve_fetal_pbtk"){
  # note that dose=0 stops the initial dosing of the compartments and is needed to accurately model the
  if(species_1 == "Rat"){
      stop("Gestation model is for human only")
  }
  if(gestationDays < 91 || gestationDays > 280) {
    print("Warning: the gestation day is the day when dosing starts, which shall be within the range of
  }

  output_list <- list()
  count=0
  for(this.cas in chemical[,"CASRN"]) {
    half_life <- calc_half_life(chem.cas = this.cas, species = species_1)
    output_f <- solve_fetal_pbtk(chem.cas = this.cas, times=seq(gestationDays,gestationDays+ndays,0.025
                                species = "human", default.to.human = TRUE, plots = FALSE, suppress.me
    output0 <- as.data.frame(output_f)
    maxAUC.h <- max(output0$AUC)*24    #convert days to hours
    maxfAUC.h <- max(output0$fAUC)*24  #convert days to hours
    concs <- output0 %>% select(c("time", "Cgut", "Cliver", "Cven", "Clung", "Cart", "Cadipose", "Crest


    if(route =="iv"){
        #ignore computational errors associated with the solver
        to_ignore <- 2 #<- time point 2 will always be anomalous
        n_to_ignore <- max(1,floor((24*ndays)/interv))
        if((24/interv) == 1){
          steps_between_doses <- 41
        }else if(24/interv > 1){
          steps_between_doses <- floor(41 / (24/interv))+1
        }else if(24/interv < 1){
          steps_between_doses <- floor(41 / (24/interv))
        }
        for(i in 1:n_to_ignore){
          to_ignore[i+1] = to_ignore[i] + steps_between_doses
        }
        concs <- concs[-to_ignore,]
    }

    #convert units and store output
    output_time.h <- sapply(concs[,1], function(x) x*24) #days to hours
    output_vals <- concs %>% select(c("Cgut", "Cliver", "Cven", "Clung", "Cart", "Cadipose", "Crest", "
    output <- as.data.frame(cbind('CASRN'=rep(this.cas, length(output_time.h)),
                                  'half_life_h' = rep(half_life, length(output_time.h)),
                                  'maxAUC.h' = rep(maxAUC.h, length(output_time.h)),
                                  'maxfAUC.h' = rep(maxfAUC.h, length(output_time.h)),
                                  output_time.h,
                                  output_vals))

    count <- count+1
    output_list[[count]] <- as.data.frame(output)
  }
}
```

Save output files

Table

An Excel output file will be created with two tabs, 1 for the PBPK parameters and one for the analysis output.

```
#build output table
#formatted_list <- lapply(output_list, assemble.table)
names(output_list) <- chemical[,"CASRN"] # <- chemical[,"CASRN"][1] for testing just one CASRN

if(modelType == "solve_pbtk"){
  formatted_list <- lapply(output_list, function(x) assemble.table_pbtk(x, nOutParam=9, nStartCol=5, nS
}
if(modelType == "solve_gas_pbtk"){
  formatted_list <- lapply(output_list, function(x) assemble.table_gas(x, nOutParam=9, nStartCol=4, nSt
}
if(modelType == "solve_fetal_pbtk"){
  formatted_list <- lapply(output_list, function(x) assemble.table_fetal(x, nOutParam=21, nStartCol=5, 
}

#for single chemical
if(length(chemical[,"CASRN"]) == 1){
  output_table <- formatted_list[[1]]
}

#for multiple chemicals
if(length(chemical[,"CASRN"]) > 1 && modelType != "solve_gas_pbtk"){
  #output_table <- rbindlist(formatted_list)
  output_table <- rbindlist(formatted_list, fill=TRUE) #to fill missing columns use fill=TRUE
  #trim redundant time rows
  to_remove <- which(output_table$compartment == 'output_time.h')[-1]
  output_table <- output_table[-to_remove]
  output_table[1,c(1:8)] <- NA
  #for solve_pbtk model, removed values for the 1st row for columns before "compartment", which are "CA
  #for solve_fetal_pbtk model,removed values for the 1st row for columns before "compartment", which ar
  colnames(output_table)[10:ncol(output_table)] <- paste('t', 1:length(output_time.h)) #"10" is where t
}

if(length(chemical[,"CASRN"])> 1 && modelType == "solve_gas_pbtk"){
    #output_table <- rbindlist(formatted_list)
    output_table <- rbindlist(formatted_list, fill=TRUE) #to fill missing columns use fill=TRUE
    to_remove <- which(output_table$compartment == 'output_time.h')[-1]#trim redundant time rows
    output_table <- output_table[-to_remove]
    output_table[1,c(1:7)] <- NA
    #removed values for the 1st row for columns before "compartment", which are "CASRN, compound, dtxsi
    colnames(output_table)[9:ncol(output_table)] <- paste('t', 1:length(output_time.h)) #"9" is where t
}
```

Preparing the chemical parameter file for writing. This will have the modeling conditions

```
param_table<-chemical
param_table$Species <-species; param_table$Model <-modelType;
param_table$"Dose, mg/kg"<-expDose;
if(route =="gas" & gasDosing =="expConc"){
  param_table$"Exposure Concentration"<-expConc            #for if inhalation
  param_table$"Exposure Concentration Unit"<-gasInputUnit  #for if inhalation
  param_table$"Exposure Length, hr"<-expLength     #for if gas
  param_table$"Exposure Period, hr"<-interv        #for if gas
  param_table<- param_table[, setdiff(colnames(param_table), c("Dose, mg/kg"))]
}

param_table$Route <- route
param_table$"Dose Interval, hrs" <- interv
param_table$"Length of Dosing, Days" <- ndays
#reformatting columns to match the other models:
param_table<-param_table %>% select(any_of(c("CASRN", "ChemicalName", "DTXSID", "Flag", "Species","Model
                               "Route","Dose, mg/kg","Exposure Concentration","Exposure Concentration Ur
                               "Clint","Clint Source", "fu", "fu Source", "MW", "LogP", "HL", "pka_Accep
colnames(param_table)<-gsub("pka_Accept", "pka Acceptor", colnames(param_table))
colnames(param_table)<-gsub("pka_Donor", "pka Donor", colnames(param_table))
```

#The output file is an excel workbook. This has 2 tabs, one for the PBPK parameters and one for the PBPK results

```
wb <- createWorkbook()
addWorksheet(wb, "PBPKresults"); writeData(wb, "PBPKresults", output_table)
addWorksheet(wb, "PBPKparameters"); writeData(wb, "PBPKparameters", param_table)
saveWorkbook(wb, file=Userout, overwrite = TRUE)
```

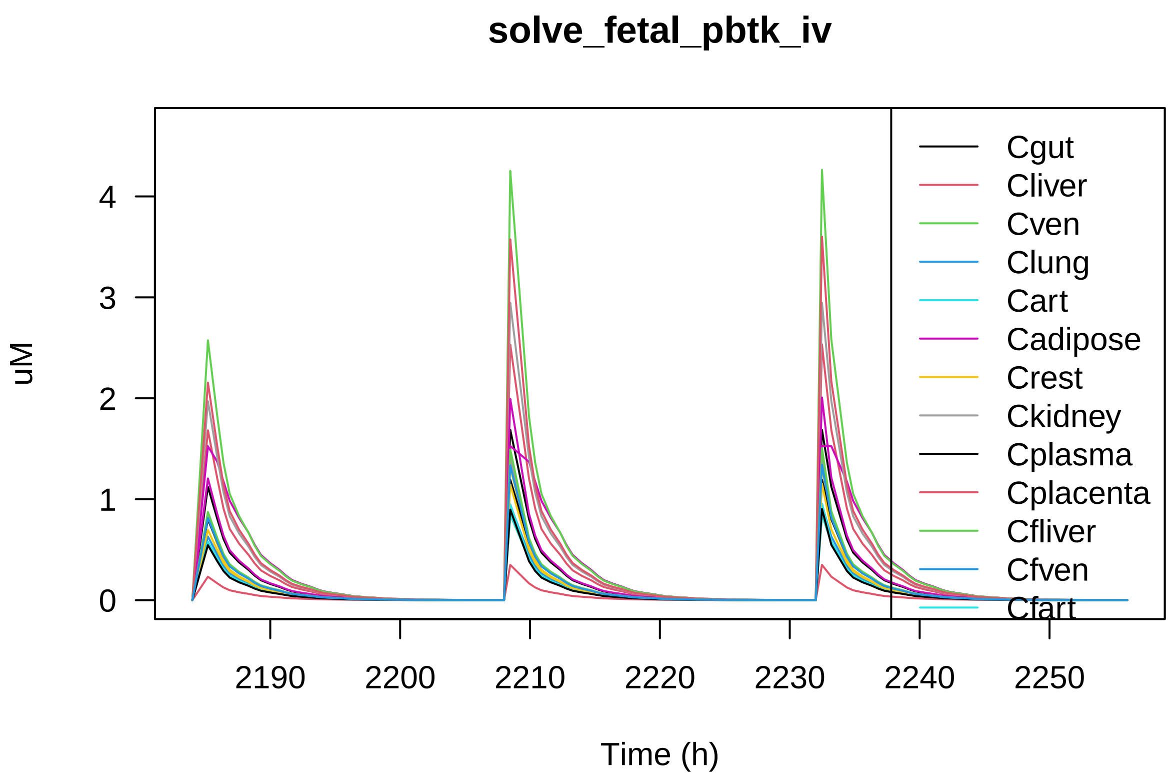
Figure

An example figure showing time series outputs for each compartment is provided. This figure shows the concentration profile in each compartment using the first CASRN provided in the chemical list.

```
example_cas_index <- which(output_table$CASRN == chemical$CASRN[1])

points_plotted <- unlist(output_table[example_cas_index, 8:ncol(output_table)])
ymax <- max(as.numeric(points_plotted),na.rm = T)*1.1

par(las=1)
plot(x=as.numeric(output_table[1,8:ncol(output_table)]),
     y=as.numeric(output_table[example_cas_index[1],8:ncol(output_table)]), type='l', ylim=c(0,ymax), yl
for(i in 2:length(example_cas_index)){
  lines(x=as.numeric(output_table[1,8:ncol(output_table)]), y=as.numeric(output_table[example_cas_index
}
legend('topright', legend = output_table$compartment[example_cas_index], lty=1, col=c(1:length(example_
```



Information on the session

```
sessionInfo()
```

```
## R version 4.3.2 (2023-10-31 ucrt)
## Platform: x86_64-w64-mingw32/x64 (64-bit)
## Running under: Windows 10 x64 (build 19045)
##
## Matrix products: default
##
##
## locale:
## [1] LC_COLLATE=English_United States.utf8
## [2] LC_CTYPE=English_United States.utf8
## [3] LC_MONETARY=English_United States.utf8
## [4] LC_NUMERIC=C
## [5] LC_TIME=English_United States.utf8
##
## time zone: America/New_York
## tzcode source: internal
##
## attached base packages:
## [1] parallel  stats     graphics  grDevices utils     datasets  methods
## [8] base
##
## other attached packages:
##  [1] data.table_1.16.0 openxlsx_4.2.7.1  httk_2.2.2        doParallel_1.0.17
##  [5] iterators_1.0.14  foreach_1.5.2     deSolve_1.40      rlang_1.1.3
##  [9] lubridate_1.9.3   forcats_1.0.0     stringr_1.5.1     dplyr_1.1.4
## [13] purrr_1.0.2       readr_2.1.5       tidyr_1.3.1       tibble_3.2.1
## [17] ggplot2_3.5.1     tidyverse_2.0.0   knitr_1.48
##
## loaded via a namespace (and not attached):
##  [1] gtable_0.3.5      xfun_0.47         lattice_0.21-9    tzdb_0.4.0
##  [5] vctrs_0.6.5       tools_4.3.2       Rdpack_2.6.1      generics_0.1.3
##  [9] fansi_1.0.6       highr_0.11        pkgconfig_2.0.3   Matrix_1.6-1.1
## [13] lifecycle_1.0.4   compiler_4.3.2    munsell_0.5.1     mitools_2.4
## [17] codetools_0.2-19  survey_4.4-2      htmltools_0.5.8.1 yaml_2.3.10
## [21] pillar_1.9.0      crayon_1.5.3      tidyselect_1.2.1  zip_2.3.1
## [25] digest_0.6.37     mvtnorm_1.3-1     stringi_1.8.3     splines_4.3.2
## [29] fastmap_1.2.0     grid_4.3.2        colorspace_2.1-1  expm_1.0-0
## [33] cli_3.6.2         magrittr_2.0.3    survival_3.5-7    utf8_1.2.4
## [37] withr_3.0.1       scales_1.3.0      bit64_4.5.2       timechange_0.3.0
## [41] rmarkdown_2.28    bit_4.5.0         hms_1.1.3         msm_1.8
## [45] evaluate_1.0.0    rbibutils_2.2.16  Rcpp_1.0.12       glue_1.6.2
## [49] DBI_1.2.3         rstudioapi_0.16.0 vroom_1.6.5       R6_2.5.1
```