

# ICE PBPK workflow

## Description

The workflow allows the flexibility to select from two different rat and human PK models from the EPA's httk package: (solve\_pbtok), and (Solve\_gas\_pbtok). The workflow is to predict plasma and tissue concentration profiles across time following a dose of a substance. Exposure routes include oral ingestion (solve\_pbtok), intravenous (solve\_pbtok), and inhalation (solve\_gas\_pbtok). For inhalation exposures using the dose is provided as an air concentration.

An example input file is included: ChemicalData\_Rnotebook.txt

## Load libraries

```
# load libraries
library(tidyverse)
library(deSolve)
library(doParallel)
library(httk) #this is needed for models: Solve_3comp, Solve_pbtok.
# 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

```
# 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(x) {
  CASRN <- rep(x$CASRN[1], 9) #8 rows, 1 for each compartment + 1 for time
  compound <- rep(chem.physical_and_invitro.data[which(chem.physical_and_invitro.data$CAS ==
    x$CASRN[1]), 1], 9)
  dtxsid <- rep(chem.physical_and_invitro.data[which(chem.physical_and_invitro.data$CAS ==
    x$CASRN[1]), 4], 9)
  Cmax <- c(NA, sapply(as.data.frame(x[4:11]), function(y) max(as.numeric(y))))
  Unit <- rep(ConcentrationUnit, 9)
  CSS_plasma <- rep(x$CSS_plasma[1], 9) #8 rows, 1 for each compartment + 1 for time
  compartment <- names(x)[3:11]
  concs <- t(x[3:11])
```

```

out <- as.data.frame(cbind(CASRN, compound, dtxsid, Cmax,
  Unit, CSS_plasma, compartment, concs))
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'
modelType <- "solve_pbtok" # 'solve_pbtok' or 'solve_gas_pbtok'

```

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 <- "oral" # 'oral', 'iv', or 'gas' needed for PBPK models.
# 'Solve_gas_pbtok' uses the inhalation (gas) route
interv <- 24 #dosing interval, hours
ndays <- 3 #number of days dosing is done
expPeriod <- 24 #length of time between exposures, hours
expDose <- 1 #current calculations assume 1mg/kg/dose
ConcentrationUnit <- "uM" #this is not available on the UI currently,
# 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 calculations assume 1mg/kg/dose
}
expLength <- 0.25 #length of the gas exposure, hours
```

## 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, ]
```

```
## 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, log10 Clint Source
## 1      1.537 OPERA
## 2      1.671 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 htk 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 source
# information
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$Clint <- 106 * chemical$Clint #this moves from log10 ul/ml/106 cells
# to just ul/ml/106 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 predictions for a
# chemical
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-05

```

## PBPK models

Currently, the PBPK analysis in ICE supports the “solve\_pbtk” and “Solve\_gas\_pbpk” model types. This is wrapper code to format the inputs and parameters needed to run the models specified by the htk package.

### Parameter processing for the PBPK models

The htk 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(htk) 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.data, data.list = list(CAS = "CASRN",
      DTXSID = "DTXSID2", Clint = "Clint", Funbound.plasma = "fu",
      pKa_Donor = "pKa_Donor", pKa_Accept = "pKa_Accept",
      logP = "LogP", logPwa = "logPwa", MW = "MW", logHenry = "HL"),
    species = species_1, reference = paste0(species_1, "ICE"),
    overwrite = T)
} else {
  chem.physical_and_invitro.data <- add_chemtable(chemical,
    current.table = chem.physical_and_invitro.data, data.list = list(CAS = "CASRN",
      DTXSID = "DTXSID2", Clint = "Clint", Funbound.plasma = "fu",
      pKa_Donor = "pKa_Donor", pKa_Accept = "pKa_Accept",
      logP = "LogP", logPwa = "logPwa", MW = "MW", logHenry = "HL"),
    species = species_1, reference = paste0(species_1, "ICE"),
    overwrite = T)
  chem.physical_and_invitro.data <- add_chemtable(chemical,
    current.table = chem.physical_and_invitro.data, data.list = list(CAS = "CASRN",
      DTXSID = "DTXSID2", Clint = "Clint", Funbound.plasma = "fu"),
    species = "Human", reference = paste0(species_1, "ICE"),
    overwrite = T) #addresses bug issues where look up from human bc assume no rat
}
if (route != "iv") {
  iv.dose = FALSE
} else {
  iv.dose = TRUE
}

dpd <- 24/interval #calculating the doses per day

```

## solve\_pbtok model

The solve\_pbtok 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_pbtok") {
  # note that dose=0 stops the initial dosing of the
  # compartments and is needed to accurately model the
  # specified dosing situation
  output_list <- list()
  count = 1
  for (this.cas in chemical[, "CASRN"]) {
    if (route == "oral") {
      concs <- solve_pbtok(chem.cas = this.cas, parameters = NULL,
        doses.per.day = dpd, days = ndays, tsteps = 4,
        dose = 0, daily.dose = expDose * dpd, iv.dose = iv.dose,
        output.units = ConcentrationUnit, species = species_1,

```

```

        default.to.human = TRUE, plots = F, suppress.messages = TRUE)[,
        c(1:10, 13)]
    } else if (route == "iv") {
      concs <- solve_pbtck(chem.cas = this.cas, parameters = NULL,
        doses.per.day = dpd, days = ndays, tsteps = 4,
        dose = 0, daily.dose = expDose * dpd, iv.dose = iv.dose,
        output.units = ConcentrationUnit, species = species_1,
        default.to.human = TRUE, plots = F, suppress.messages = TRUE)[,
        c(1:10, 13)]
      # 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.per.day = dpd,
      species = species_1, model = "pbtck", output.units = ConcentrationUnit)

    # convert units and store output
    output_time.h <- sapply(concs[, 1], function(x) x * 24) #days to hours
    output_vals <- concs[, 3:11]
    output <- as.data.frame(cbind(CASRN = rep(this.cas, length(output_time.h)),
      CSS_plasma = rep(CSS0, length(output_time.h)), output_time.h,
      output_vals))

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

```

## solve\_gas\_pbtck model

The solve\_gas\_pbtck is a PBPK model similar to the “solve\_pbtck” model but is uses an inhalation route of exposure assuming the chemical is volatile (gas). As a result, it has some additional checks to see if the assumption of gas exposure is reasonable but will still proceed with a flag warning.

```

if (modelType == "solve_gas_pbtck") {
  # check the assumption of volatility
  chemical$Flag[chemical$HL <= -7.80388 & chemical$Flag !=

```

```

    """] <- "Fu is zero, likely due to rounding. Setting to 1e-5;
    Chemical likely nonvolatile, consider appropriateness of model"
chemical$Flag[chemical$HL <= -7.80388 & chemical$Flag ==
    """] <- "Chemical likely nonvolatile, consider appropriateness of model"

# note that dose=0 stops the initial dosing of the
# compartments and is needed to accurately model the
# specified dosing situation
output_list <- list()
count = 1
for (this.cas in chemical[, "CASRN"]) {

  # --- expDose option has currently been disabled due to a
  # bug in the httk package --- if(gasDosing == 'expDose'){
  # ConcentrationUnit <- 'uM' # output unit only has one option
  # of 'uM' for Solve_gas_pbt model concMax <-
  # max(solve_gas_pbt(chem.cas = this.cas, parameters = NULL,
  # doses.per.day = dpd, days = ndays, tsteps = 4, dose=0,
  # daily.dose = expDose*dpd, exp.conc = 0, period=expPeriod,
  # exp.duration = expLength, output.units = ConcentrationUnit,
  # species = species_1, default.to.human = TRUE, plots = F,
  # suppress.messages = TRUE)[, 'Cplasma']) }

  if (gasDosing == "expConc") {
    # output unit only has one option of 'uM' for Solve_gas_pbt
    # model
    ConcentrationUnit <- "uM"
    concs <- solve_gas_pbt(chem.cas = this.cas, parameters = NULL,
      doses.per.day = dpd, days = ndays, tsteps = 4,
      dose = 0, daily.dose = 0, exp.conc = expConc,
      period = expPeriod, exp.duration = expLength,
      output.units = ConcentrationUnit, species = species_1,
      default.to.human = TRUE, plots = F, suppress.messages = TRUE)[,
      c(1:10, 17)]
  }

  # steady state concentration
  CSS0 <- calc_analytic_css(chem.cas = this.cas, parameters = NULL,
    daily.dose = expDose * dpd, doses.per.day = dpd,
    species = species_1, model = "pbt", output.units = ConcentrationUnit)

  # convert units and store output
  output_time.h <- sapply(concs[, 1], function(x) x * 24) #days to hours
  output_vals <- concs[, 3:11]
  output <- as.data.frame(cbind(CASRN = rep(this.cas, length(output_time.h)),
    CSS_plasma = rep(CSS0, length(output_time.h)), output_time.h,
    output_vals))

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

```

## 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)
output_table <- rbindlist(formatted_list)

# trim redundant time rows
to_remove <- which(output_table$compartment == "output_time.h")[-1]
output_table <- output_table[-to_remove]
output_table[1, c(1:6)] <- NA

colnames(output_table)[8:ncol(output_table)] <- paste("t", 1:length(output_time.h))

# preparing the chemical parameter file for writing. This
# will have the modeling conditions
param_table <- chemical
param_table$Species <- species
param_table$Model <- modelType
if (route == "gas" & gasDosing == "expConc") {
  param_table$"Exposure Concentration,uM/L" <- expConc #for if inhalation
  param_table$"Exposure Length, hr" <- expLength #for if inhalation
  param_table$"Exposure Period, hr" <- expPeriod #for if inhalation
}

param_table$"Dose, mg/kg" <- expDose
param_table$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,uM/L", "Exposure Length, hr", "Exposure Period, hr",
  "Dose Interval, hrs", "Length of Dosing, Days", "Clint",
  "Clint Source", "fu", "fu Source", "MW", "LogP", "HL", "pka_Accept",
  "pka_Donor", "logPwa"))))
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)
```



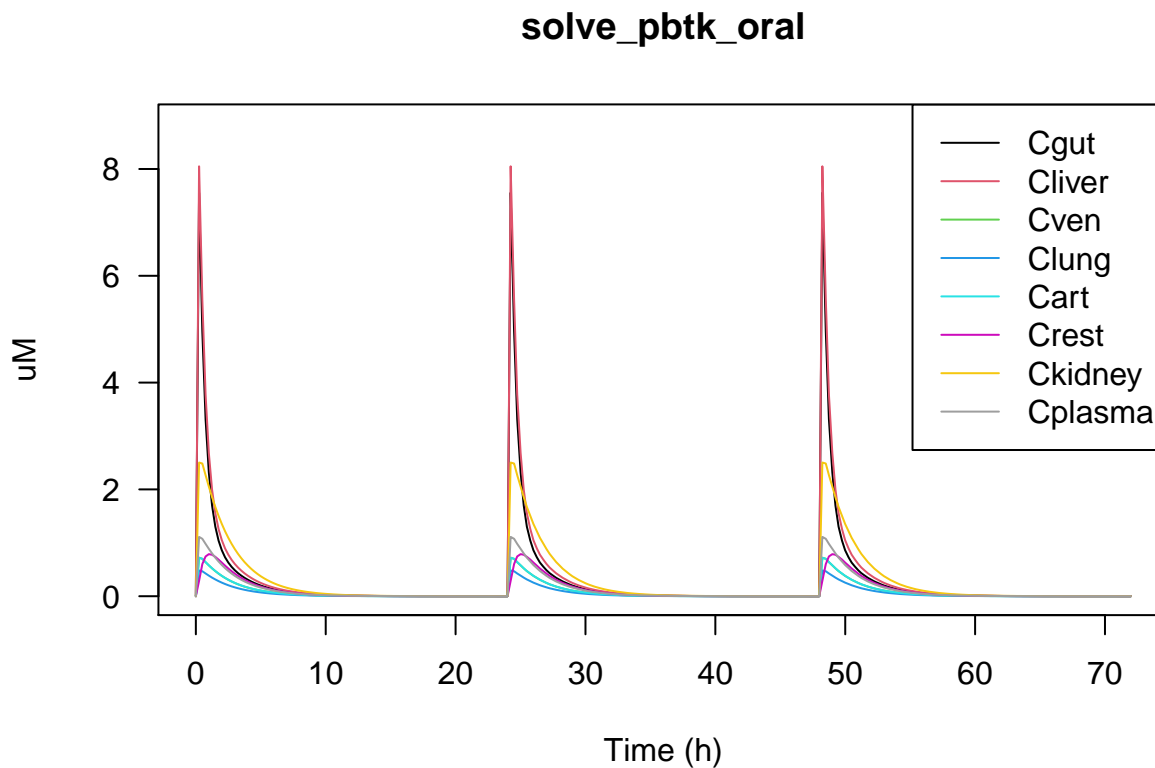
## Figure

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))*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), ylab = ConcentrationUnit, xlab= 'Time (h)',
     main=paste(modelType, '_', route, sep=''))
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[i],8:ncol(output_table)]), col=i)
}
legend('topright', legend = output_table$compartment[example_cas_index],
      lty=1, col=c(1:length(example_cas_index)))
```



## Information on the session

```
sessionInfo()
```

```
## R version 4.0.2 (2020-06-22)
## Platform: x86_64-w64-mingw32/x64 (64-bit)
## Running under: Windows 10 x64 (build 17763)
##
## Matrix products: default
##
## locale:
## [1] LC_COLLATE=English_United States.1252
## [2] LC_CTYPE=English_United States.1252
## [3] LC_MONETARY=English_United States.1252
## [4] LC_NUMERIC=C
## [5] LC_TIME=English_United States.1252
##
## attached base packages:
## [1] parallel stats graphics grDevices utils datasets methods
## [8] base
##
## other attached packages:
## [1] data.table_1.13.6 openxlsx_4.2.3 httpk_2.0.3 doParallel_1.0.16
## [5] iterators_1.0.13 foreach_1.5.1 deSolve_1.28 forcats_0.5.1
## [9] stringr_1.4.0 dplyr_1.0.4 purrr_0.3.4 readr_1.4.0
## [13] tidyr_1.1.2 tibble_3.0.4 ggplot2_3.3.3 tidyverse_1.3.0
## [17] knitr_1.31
##
## loaded via a namespace (and not attached):
## [1] Rcpp_1.0.5 mvtnorm_1.1-1 msm_1.6.8 lubridate_1.7.9.2
## [5] lattice_0.20-41 assertthat_0.2.1 digest_0.6.27 R6_2.5.0
## [9] cellranger_1.1.0 backports_1.2.1 reprex_1.0.0 survey_4.0
## [13] evaluate_0.14 highr_0.8 httr_1.4.2 pillar_1.4.7
## [17] rlang_0.4.10 readxl_1.3.1 rstudioapi_0.13 Matrix_1.2-18
## [21] rmarkdown_2.6 splines_4.0.2 munsell_0.5.0 broom_0.7.4
## [25] compiler_4.0.2 modelr_0.1.8 xfun_0.20 pkgconfig_2.0.3
## [29] htmltools_0.5.1.1 mitools_2.4 tidyselect_1.1.0 expm_0.999-6
## [33] codetools_0.2-16 crayon_1.4.0 dbplyr_2.0.0 withr_2.4.1
## [37] grid_4.0.2 jsonlite_1.7.2 gtable_0.3.0 lifecycle_0.2.0
## [41] DBI_1.1.1 magrittr_2.0.1 formatR_1.8 scales_1.1.1
## [45] zip_2.1.1 cli_2.3.0 stringi_1.5.3 fs_1.5.0
## [49] xml2_1.3.2 ellipsis_0.3.1 generics_0.1.0 vctrs_0.3.6
## [53] tools_4.0.2 glue_1.4.2 hms_1.0.0 survival_3.1-12
## [57] yaml_2.2.1 colorspace_1.4-1 rvest_0.3.6 haven_2.3.1
```