ICE IVIVE workflow

Description

The workflow allows the flexibility to select from three different rat and human PK models: a 1 compartment model that incorporates Monte Carlo simulation to simulate the population variance (1C), a 3 compartment model leveraging the EPA's httk package (Solve_3comp), (solve_pbtk), and (Solve_gas_pbtk). The workflow is to predict the daily equivalent administered dose (EAD, mg/kg/dose) that would lead to steady state blood concentration equivalent to the bioactive concentration from in vitro assays. For inhalation exposures using the (Solve_gas_pbtk) EADs are generated in uM if using "concentration" exposure vs a dose. 2 example files are included: * ChemicalData_Rnotebookv3.2.txt * InvitroData_Rnotebookv3.2.txt

Load libraries

```
# load libraries
library(tidyverse)
library(deSolve)
library(doParallel)
library(httk)  #this is needed for models: solve_3comp, solve_pbtk.
# The code is compatible with httk_2.0.2
library(xlsx)  #for writing the excel file that is user output
```

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 assayFile: the file with the in vitro bioactivity data. The first column is the CASRN and the subsequent columns are the bioactivity values output_file: this is the file for the outputs

```
chemFile <- "ChemicalData_Rnotebook.txt"  #chemicals data from ICE, include CASRN
# field as identifier
assayFile <- "InvitroData_Rnotebook.txt"  #invitro data from ICE, includes CASRN
# field as identifier then acc/ac50
Userout <- "User_Results.xlsx"  #file name (and path) for the output file for the tissue
# concentrations with EAD estimations along with the in vitro
# data
EADplot_file <- "User_plot.pdf"  #the file for the plot of the EAD plot

invivo <- NULL  #invivo data provided by user
# invivoFile <- 'InvivoData_Rnotebook.txt' invivo <-
# as.data.frame(read_delim(invivoFile, delim = '\t'))
```

Model variables

Details about what model, route, and dose need to be specified. There are differences in what is needed if a 1 compartment model is used vs the PBPK models

What model?

Models type is limited to 4 different models. 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 <- "1C"  #'1C', solve_3comp', 'solve_pbtk', 'solve_gas_pbtk'
```

For the 1 compartment model, values are needed to parameterize the Monte Carlo simulation.

```
nsamples <- 300  #user-provided value for the mc simulations, any number between 10 - 10,000
```

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 and route, interval, and days are needed for all PBPK models.

```
route <- "iv"  #oral, iv, or inhalation needed for PBPK models.
#'Solve_gas_pbtk' uses the inhalation route
interv <- 24  #dosing interval, hours
ndays <- 3  #number of days dosing is done
# expPeriod<-12 #length of time between exposures, hours
expDose <- 1  #current calculations assume 1mg/kg/dose for '1C', 'solve_3comp', 'solve_pbtk';
ConcentrationUnit <- "uM"  #this is not available on the UI currently, options are uM and mg/L
chemDisplay <- "ChemicalName"  #Argument for EADboxplot.R, choice is 'CASRN' or 'ChemicalName'
EADUnit <- "mg/kg/dose"  #Argument for EADboxplot.R, 'mg/kg/dose' for '1C', 'solve_3comp', 'solve_pbtk'
# uM for 'solve_gas_pbpk' model
```

##Inhalation specific parameters For an inhalation route of exposure, there are 2 different ways that the exposure can be modeled. "expDose" models a set bolus dose like the exposure for an oral or IV exposure. The "expConc" is more typical of a gas exposure. This models an inhaled concentration over a duration of time, set by "expLength". The "expDose" option is currently unavailable.

```
gasDosing <- "expConc"  #two dosing option methods for Solve_gas_pbtk model,
#'expConc' and 'expDose',
# but for ICE3.3 release, 'expConc' is the only option due to
# a bug found in httk function This is only needed if
# gasDosing =='expConc' because otherwise default to expDose
if (gasDosing == "expConc" & (!exists("expConc") || is.null(expConc))) {
    expConc <- 1  #current calculations assume 1uM of air concentration
}
expLength <- 0.25  #length of the gas exposure, hours
```

Load functions

```
# All required R scripts and input files should be in the
# working directory
source("steadyState.R")  #required for '1C' model
source("CalcEAD.R")  #required for '1C', 'solve_3comp', 'solve_pbtk', 'solve_gas_pbpk' models
source("EADboxplot.R")
```

Load data

Notice that the chem 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.

```
# there are single quotes in chemical names that need to be
# addressed, the tidyR handels well
chemical <- as.data.frame(read_delim(chemFile, delim = "\t"))
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
```

```
invitro <- as.data.frame(read_delim(assayFile, delim = "\t"))
invitro[1:2, ]
```

```
##        CASRN ACEA_ER_80hr    ATG_Ahr_CIS_up    ATG_GRE_CIS_up
## 1  4419-39-0             NA                NA                NA
## 2 74050-98-9             NA                NA                NA
##   OT_ER_ERaERb_0480    OT_ER_ERaERb_1440    OT_ER_ERbERb_0480
## 1                NA                   NA                   NA
## 2                NA                   NA                   NA
##   OT_ER_ERbERb_1440    TOX21_AhR_LUC_Agonist    TOX21_CAR_Agonist
## 1                NA                       NA                   NA
## 2                NA                       NA                   NA
##   TOX21_ERb_BLA_Antagonist_ratio    TOX21_GR_BLA_Agonist_ratio
## 1                             NA                         0.033
## 2                             NA                            NA
##   TOX21_GR_BLA_Antagonist_ratio    TOX21_TR_LUC_GH3_Agonist
## 1                            NA                          NA
## 2                            NA                          NA
```

```
##   TOX21_TR_LUC_GH3_Antagonist    TOX21_TSHR_Agonist_ratio
## 1                          NA                          NA
## 2                          NA                          NA
##   TOX21_TSHR_Antagonist_ratio
## 1                          NA
## 2                          NA
```

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 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 this moves
# from log10 ul/ml/10^6 cells to just ul/ml/10^6 cells
chemical$Clint <- 10^chemical$Clint
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
```

1C modeling

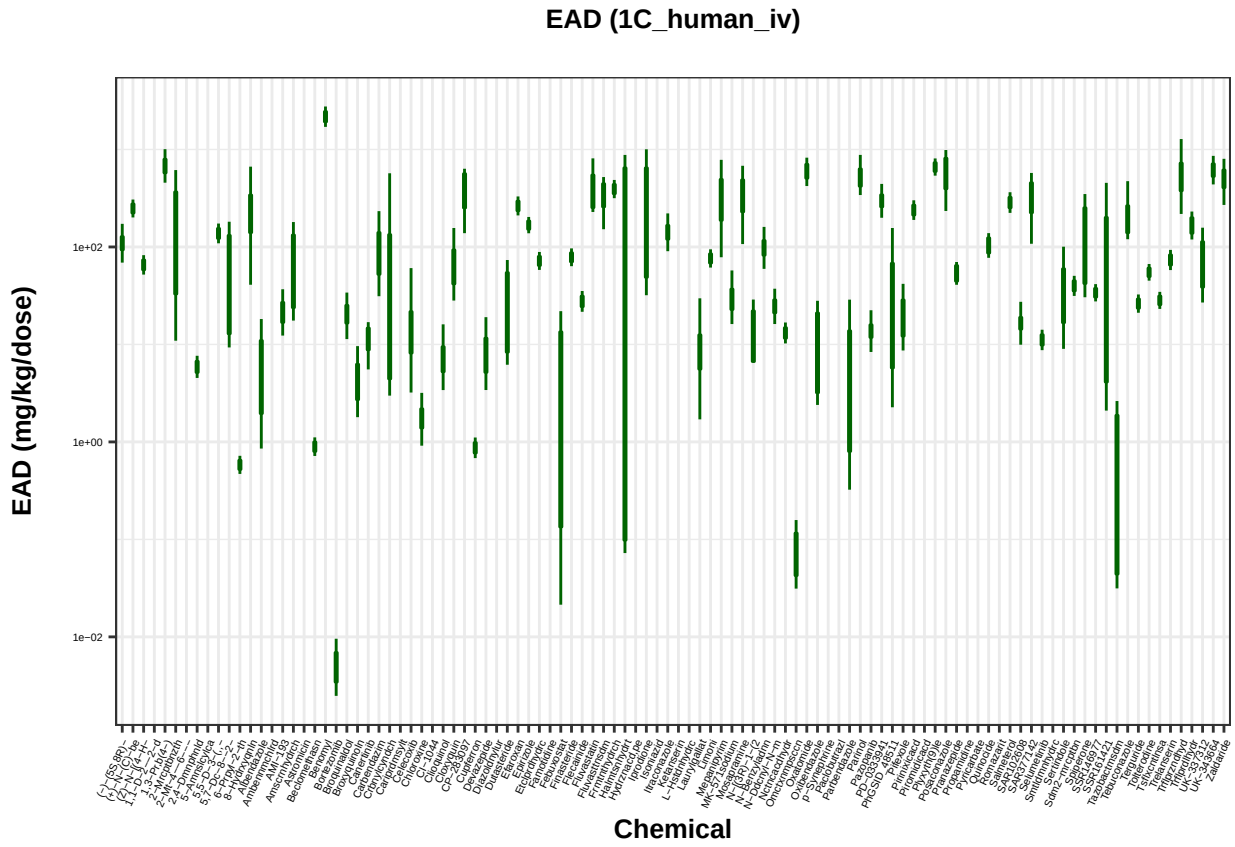
If modelType given by the user is "1C" This will run a simple, 1 compartment model. In this model the C steady state is determined vs the C max from the other models.

Calculating the EADs

The 1C model has a monte carlo simulation that calculates the Css at the 50th and the 95th percentile The first step is generating the CSS object with the call to "steadState".

```
### model '1C'
if (modelType == "1C") {

    chemInput <- chemical[, c("CASRN", "ChemicalName", "Clint",
        "fu", "MW", "Flag")]
    CSS <- steadyState(inputData = chemInput, nsamples = nsamples,
        species = species, ConcentrationUnit = ConcentrationUnit)
    head(CSS)

    # There are 2 different steady state predictions, 50% and 95%
    # ile. The EAD is calculated for each using the 'CalcEAD'
    # function. The data is then formated into a single data
    # object

    EAD.out50 <- CalcEAD(Css = CSS[, c("CASRN", "50%", "fu")],
        inVitro = invitro, adj.fu = "fu")
    colnames(EAD.out50) <- gsub("EAD", "EAD.50", colnames(EAD.out50))
    # add in the flag column
    EAD.out50 <- left_join(chemInput[, c("CASRN", "Flag")], EAD.out50)
    EAD.out95 <- CalcEAD(Css = CSS[, c("CASRN", "95%", "fu")],
        inVitro = invitro, adj.fu = "fu")
    colnames(EAD.out95) <- gsub("EAD", "EAD.95", colnames(EAD.out95))
    EAD.out <- left_join(EAD.out50, EAD.out95)
    EAD.out <- EAD.out[, setdiff(colnames(EAD.out), c("adj.fu",
        "fu", "adj.arm", "arm"))]  # remove the columns 'fu', adj.fu','adj.arm' and 'arm'
    # creating output file for user, should have EAD values and
    # the parameters for calculating the circulating
    # concentration
    CSS2 <- as.data.frame(CSS, stringsAsFactors = FALSE)
    colnames(CSS2) <- gsub("50%", "Css, 50%ile", colnames(CSS2))
    colnames(CSS2) <- gsub("95%", "Css, 95%ile", colnames(CSS2))
    CSS2$Species <- species
    CSS2$Model <- modelType
    CSS2$"Dose, mg/kg" <- expDose
    CSS2$Route <- route
    CSS2$nSimulations <- nsamples
    colnames(CSS2) <- gsub("Css_Unit", "Css, Units", colnames(CSS2))

    CSS2 <- left_join(CSS2, chemical %>% select(any_of(c("CASRN",
        "Clint Source", "fu Source"))))

    # reformatting columns to match the other models:
    CSS2 <- CSS2 %>% select(any_of(c("CASRN", "ChemicalName",
```

```
        "Css, 50%ile", "Css, 95%ile", "Css, Units", "Flag", "Species",
        "Model", "Route", "Clint", "Clint Source", "fu", "fu Source",
        "MW")))
    ssEAD.out <- EAD.out[, c("CASRN", setdiff(colnames(EAD.out),
        c(colnames(invitro), "50%", "95%")))]
    outputData <- full_join(CSS2, ssEAD.out)
    colnames(outputData) <- gsub("Clint$", "Clint, ul/ml/10^6 cells",
        colnames(outputData))  #add units

    # The output file is an excel workbook. This has 2 tabs, one
    # for the EAD results and the other for the invitro data
    xlsx::write.xlsx(file = Userout, x = outputData, sheetName = "EADResults",
        append = FALSE, row.names = FALSE, showNA = FALSE)  #starting a new book
    xlsx::write.xlsx(file = Userout, x = invitro, sheetName = "inVitroData",
        append = TRUE, row.names = FALSE, showNA = FALSE)

    # To view the results, the 'EADboxplot' function plots the
    # values.
    EADplot <- EADboxplot(EAD.out = outputData, invivo = invivo,
        label = "EAD", EADUnit = EADUnit, species = species,
        route = route, modelType = modelType, chemDisplay = chemDisplay,
        axis.text = 4)
    # Arguments for EADplot functions include 'EAD.out, invivo,
    # label, chemDisplay, EADUnit, modelType, species, route,
    # date, doses.per.day, wth, axis.text, title.text,
    # shape.size, dpi'
    EADplot
}
```



PBPK models

This code is used if modelType given by the user is a PBPK model: “solve_3comp”, “solve_pbtk”, or “Solve_gas_pbpk”. This is wrapper code to format the inputs and parameters needed to run the models specified by the httk package. In addition, the output is formatted so that it matches the 1C model. This allows easy comparison between the different files for subsequent processing.

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.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/interv  #calculating the doses per day
```

solve_3comp model

The solve_3comp model is a 3 compartment model that has the gut,gut lumen, liver, and rest of the body compartments with the plasma equivalent to the liver plasma concentrations.

```
if (modelType == "solve_3comp") {
    cmaxall <- NULL
    # note that dose=0 stops the initial dosing of the
    # compartments and is needed to accurately model the
    # specified dosing situation
    for (this.cas in chemical[, "CASRN"]) {
        concMax <- max(solve_3comp(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)[, "Cplasma"])
        cmax_temp <- as.data.frame(cbind(this.cas, concMax, ConcentrationUnit))
        cmaxall <- rbind(cmaxall, cmax_temp)
    }
}
```

solve_pbtk model

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

```
if (modelType == "solve_pbtk") {
    cmaxall <- NULL
    # note that dose=0 stops the initial dosing of the
    # compartments and is needed to accurately model the
    # specified dosing situation
    for (this.cas in chemical[, "CASRN"]) {

        if (route == "oral") {
            concMax <- max(solve_pbtk(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)[,
                "Cplasma"])
        } else if (route == "iv") {
            concs <- solve_pbtk(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)[,
                "Cplasma"]
            # 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
            }
            concMax <- max(concs[-to_ignore])
        }

        cmax_temp <- as.data.frame(cbind(this.cas, concMax, ConcentrationUnit))
        cmaxall <- rbind(cmaxall, cmax_temp)
    }
}
```

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 volitle (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. It also includes 2 different dosing approaches, a concentration over time (expConc) or a single dose (expDose)

```
if (modelType == "solve_gas_pbtk") {
    gasDosing == "expConc"  #ensure gas dosing is set to expConc
    # check the assumption of volitility
    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"

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

        # --- expDose option has currently been disabled due to a bug
        # in the httk package --- if(modelType=='solve_gas_pbtk' &
        # 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, days = ndays,
        # tsteps = 4, dose=0, daily.dose = expDose*dpd, exp.conc = 0,
        # period=interv, exp.duration = expLength, output.units =
        # ConcentrationUnit, species = species_1, default.to.human =
        # TRUE, plots = F, suppress.messages = TRUE)[,'Cplasma']) }

        if (modelType == "solve_gas_pbtk" & gasDosing == "expConc") {
            # output unit only has one option of 'uM' for Solve_gas_pbtk
            # model
            ConcentrationUnit <- "uM"
            concMax <- max(solve_gas_pbtk(chem.cas = this.cas,
                parameters = NULL, doses.per.day = dpd, days = ndays,
                tsteps = 4, dose = 0, daily.dose = 0, exp.conc = expConc,
                period = interv, exp.duration = expLength, output.units = ConcentrationUnit,
                species = species_1, default.to.human = TRUE,
                plots = F, suppress.messages = TRUE)[, "Cplasma"])
        }
        cmax_temp <- as.data.frame(cbind(this.cas, concMax, ConcentrationUnit))
        cmaxall <- rbind(cmaxall, cmax_temp)
    }
}
```

Calc EAD from PBPK

The PBPK models currently give estimates based on the median (50%ile) of the population. As such there is minor formatting that needs to be done to generate the EAD predictions and make the output file

```
if (modelType == "solve_3comp" | modelType == "solve_pbtk" |
    modelType == "solve_gas_pbtk") {
    cmaxall$concMax <- as.numeric(cmaxall$concMax)
```

```
# they are bound as characters so converting to numeric

Cmax <- merge(chemical, cmaxall, by.x = "CASRN", by.y = "this.cas")
names(Cmax) <- gsub("concMax", "Cmax", names(Cmax))
# Calculating the EADs
EAD.out_max <- CalcEAD(Css = Cmax[, c("CASRN", "Cmax")],
    inVitro = invitro)
# creating output file for user, should have EAD values and
# the parameters for calculating the circulating
# concentration
CSS2 <- as.data.frame(Cmax, stringsAsFactors = FALSE)
# adding in the source of the parameters
CSS2 <- left_join(CSS2, chemical %>% select(any_of(c("CASRN",
    "Clint Source", "fu Source"))))
CSS2$Species <- species
CSS2$Model <- modelType
CSS2$"Dose, mg/kg" <- expDose
ssEAD.out <- EAD.out_max[, c("CASRN", setdiff(colnames(EAD.out_max),
    c(colnames(invitro), "Cmax")))]
colnames(CSS2) <- gsub("ConcentrationUnit", "Cmax, Units",
    names(CSS2))
CSS2$Route <- route
CSS2$"Dose Interval, hrs" <- interv
CSS2$"Length of Dosing, Days" <- ndays
# reformatting columns to match the other models:
CSS2 <- CSS2 %>% select(any_of(c("CASRN", "ChemicalName",
    "Cmax", "Cmax, Units", "Flag", "Species", "Model", "Route",
    "Dose Interval, hrs", "Length of Dosing, Days", "Clint",
    "Clint Source", "fu", "fu Source", "MW")))

outputData <- full_join(CSS2, ssEAD.out)
colnames(outputData) <- gsub("Clint$", "Clint, ul/ml/10^6 cells",
    colnames(outputData))  #add units

# The output file is an excel workbook. This has 2 tabs, one
# for the EAD results and the other for the invitro data

xlsx::write.xlsx(file = Userout, x = outputData, sheetName = "EADResults",
    append = FALSE, row.names = FALSE, showNA = FALSE)  #starting a new book
xlsx::write.xlsx(file = Userout, x = invitro, sheetName = "inVitroData",
    append = TRUE, row.names = FALSE, showNA = FALSE)

# to view the results, the 'EADboxplot' function plots the
# values.  Note that if the gas model is used EADUnit needs
# to be changed to 'uM'

# plotting
EADplot <- EADboxplot(EAD.out = outputData, invivo = invivo,
    label = "EAD", EADUnit = EADUnit, species = species,
    route = route, modelType = modelType, doses.per.day = dpd,
    chemDisplay = chemDisplay, axis.text = 4)
# Arguments for EADplot functions include 'EAD.out, invivo,
# label, chemDisplay, EADUnit, modelType, species, route,
```

```
    # date, doses.per.day, wth, axis.text, title.text,
    # shape.size, dpi'
    EADplot
}
```

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] scales_1.1.1      xlsx_0.6.5        httk_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   xlsxjars_0.6.1    assertthat_0.2.1  digest_0.6.27
##  [9] R6_2.5.0          cellranger_1.1.0  backports_1.2.1   reprex_1.0.0
## [13] survey_4.0        evaluate_0.14     highr_0.8         httr_1.4.2
## [17] pillar_1.4.7      rlang_0.4.10      readxl_1.3.1      rstudioapi_0.13
## [21] data.table_1.13.6 Matrix_1.2-18     rmarkdown_2.6     splines_4.0.2
## [25] munsell_0.5.0     broom_0.7.4       compiler_4.0.2    modelr_0.1.8
## [29] xfun_0.20         pkgconfig_2.0.3   htmltools_0.5.1.1 mitools_2.4
## [33] tidyselect_1.1.0  expm_0.999-6      codetools_0.2-16  crayon_1.4.0
## [37] dbplyr_2.0.0      withr_2.4.1       grid_4.0.2        jsonlite_1.7.2
## [41] gtable_0.3.0      lifecycle_0.2.0   DBI_1.1.1         magrittr_2.0.1
## [45] formatR_1.8       cli_2.3.0         stringi_1.5.3     farver_2.0.3
## [49] fs_1.5.0          xml2_1.3.2        ellipsis_0.3.1    generics_0.1.0
## [53] vctrs_0.3.6       tools_4.0.2       glue_1.4.2        hms_1.0.0
## [57] survival_3.1-12   yaml_2.2.1        colorspace_1.4-1  rvest_0.3.6
## [61] rJava_0.9-13      haven_2.3.1
```