pksensi: A Package to Apply Global Sensitivity Analysis in Physiologically Based Kinetic Modeling

by Nan-Hung Hsieh, Brad Reisfeld, Weihsueh A. Chiu

Abstract We developed a package, named **pksensi**, to make global sensitivity analysis (SA) more accessible in physiologically based kinetic modeling. This package can investigate both parameter uncertainty and sensitivity in pharmacokinetic models, including those with multivariate model outputs, such as multiple time-points and tissue compartments. We refined the extended Fourier Amplitude Sensitivity Test method for SA by adding a random phase-shift to analyze the statistical variability of the sensitivity index for each model parameter. Furthermore, it includes functions to check the convergence of the global SA results. Utilizing **pksensi**, we successfully reproduced our previously published SA results of human physiologically based pharmacokinetic modeling for acetaminophen and its two primary metabolites. Overall, **pksensi** improves the user experience of performing global SA and can create robust and reproducible results for decision making in pharmacokinetic model calibration.

Introduction

Sensitivity analysis (SA) is a mathematical technique to investigate how variations in model parameters affect model outputs. An increasing number of studies use SA to determine which model parameters contribute to high variation in model predictions (Ferretti et al., 2016). This technique has also been applied in pharmacology and toxicology research (Loizou et al., 2015; McNally et al., 2012). Pharmacokinetic (PK) models describe the changes in the concentrations or amounts of a substance in the body over time (the various terms for the kinetic models, including "pharmacokinetic", "toxicokinetic", and "biokinetic" models, are used exchangeably here). The goal of SA in PK research is to examine the sensitivity of output variables, such as chemical concentration in blood or tissues, with respect to input parameters, such as anatomical, physiological, and kinetic constants (McNally et al., 2011). In addition, this SA approach can guide experimental design and the parameter estimation processes (Zhang et al., 2015). For instance, SA can identify "unidentifiable" parameters that can lead to problems with numerical procedures used in parameter estimation (Chu and Hahn, 2010). It can be further applied to parameter prioritization and parameter fixing before model calibration (Hsieh et al., 2018).

In our earlier work (Hsieh et al., 2018), we developed an updated workflow to apply global SA to reduce the computational burden in the Bayesian, Markov Chain Monte Carlo (MCMC)-based calibration process of a physiologically based pharmacokinetic (PBPK) model. We used GNU MCSim (Bois, 2009), an effective simulation package for Bayesian population PBPK modeling, to calibrate the model. We found that the extended Fourier Amplitude Sensitivity Test (eFAST), a type of variance-based global SA algorithm, had the best balance of efficiency and accuracy for a sophisticated, multi-compartment, multi-dataset, and multi-metabolite PBPK model. This method was first introduced in Cukier et al. (1973) (the classic FAST method with the first-order effect only) in a study of the sensitivity of coupled reaction systems that are constructed by differential equations with the numerous rate coefficients. It then evolved to estimate the Sobol' sensitivity measure in Saltelli et al. (1999) (the eFAST method with first- and total-order effects). The FAST algorithm creates the multidimensional space of input parameters through a "search-curve". It is more computationally efficient in calculating the influence of model parameter than other global SA methods such as the Monte Carlo based approaches (Jansen, 1999; Owen, 2013). Our previous work also found some efficient visualization approaches that can be used to distinguish between "influential" and "non-influential" parameters. This approach addresses common identifiability issues of PBPK model parameter that increase the computational burden of Bayesian analysis (Garcia et al., 2015), and therefore reduce the computational efficiency. We also developed approaches for communicating parameter sensitivity in decision making.

There are numerous developed packages, such as **sensitivity** (Iooss et al., 2018), which is a practical tool to conduct local and global SA. The **sensitivity** package includes some functions to generate the parameter sequences by using different computing algorithms. Also, it provides a feasible way to integrate external modeling results. This computational approach can effectively solve the heavy computing burden of numerical solutions within the pure R environment. Several other R packages, such as **FME**, **multisensi**, and **ODEsensitivity** include SA tools and are freely available in the Comprehensive R Archive Network (CRAN). The **FME** package contains a basic method for performing local SA for dynamic models (Soetaert and Petzoldt, 2010). It can also conduct collinearity

and identifiability analysis to evaluate which model parameters can estimate or exclude based on available observation data. The **multisensi** package provides SA for models with multivariate output (Bidot et al., 2018). **ODEsensitivity** can perform SA in ordinary differential equation (ODE) models (Weber and Theers, 2019). It utilizes the ODE interface from the **deSolve** package and connect it with the SA from the **sensitivity** package. Other related packages such as **mtk** (Wang et al., 2015) and **fast** (Reusser, 2015) are designed to investigate the importance of model parameter.

Although these tools provide various approaches to perform SA, they each individually have limitations for application to PK modeling. For example, the estimated index in SA is not robust when using the inadequate size of the sample number. However, there are no software packages that provide the available tools in the assessment of reliability. In addition, the existing eFAST function does not include the feature to generate the random sampling curve. We present here an package, called **pksensi**, which is designed to make SA more accessible and reproducible in pharmacological and toxicological researches. This package can investigate both parameter uncertainty and sensitivity in PK model with multivariate model outputs. Moreover, it seamlessly integrates with both general ODE solvers available in R or as external C code, as well as the GNU MCSim software used for conducting sophisticated PK modeling, including MCMC. The analytical framework in **pksensi** includes not only global SA but also the uncertainty analysis and diagnostic tools to support the modeling process.

Workflow

Installation

The **pksensi** package can link with **deSolve** package, which includes a comprehensive integrator to solve ODE. The GNU MCSim can be installed by following the instruction in GNU MCSim's manual on `https://www.gnu.org/software/mcsim/mcsim.html` or using the built-in function `mcsim_install` in **pksensi**. The C compiler is already available in Linux or MacOS. The Windows user can install Rtools and set the working path of compiler before using this function as:

```
Rtools.path <- "c:/Rtools/bin; c:/Rtools/mingw_64/bin"
Sys.setenv(PATH = paste(Rtools.path, Sys.getenv("PATH"), sep=";"))
```

Parameter matrix generation

We adopted eFAST, the widely used analysis of variance (ANOVA)-like global SA approach to decompose the variance to partial variances for multidimensional parameters in numerical models in this package. Most of the available eFAST functions, such as the algorithm to generate the parameter space and to set the sampling frequency, were sourced from the **sensitivity** package (Iooss et al., 2018). The eFAST method computes the sensitivity index (SI) via ANOVA-like decomposition of the function for analysis. This type of global SA approach and SI also know as Sobol' method and Sobol' index. Since the estimated SI is not stable (high variation) when using smaller sample size, we included a random phase-shift approach to conduct replicating sampling from random starting points across parameter space to test the convergence and robustness of the sensitivity measurement. Considering the model that is built under $y = f(x_i)$. The sampling scheme can describe as,

$$x_i = \frac{1}{2} + \frac{1}{\pi} \arcsin(\sin(\omega_i s + \varphi_i))$$

where x_i is the nominal value of the i-th parameter, and ω_i is a vector giving the set of frequencies, one frequency for each parameter, and φ is a random phase-shift coefficient ranged from 0 to 2π. The default set of frequencies is according to the study from Saltelli et al. (1999). Through the random phase-shift, the robust result of the sensitivity measurement should be similar across each replication under the same sample size. For most Sobol'-based SA algorithms, the larger sample size is needed to reduce the probability of having negative estimates. However, FAST estimates are always positive according to its construction.

To investigate the convergence of SI, we adopted the "exam" approach proposed by (Sarrazin et al., 2016). This method quantitatively assesses convergence by computing the width of 95% confidence intervals of computed SI for all parameters across all time-points and output variables. Through random phase-shift procedure, the replicated SI values can repeatedly sample independently. Therefore, they can be used to investigate the convergence of SI as,

$$CI_{i,t} = max(SI_{i,t}^{ub} - SI_{i,t}^{lb})$$

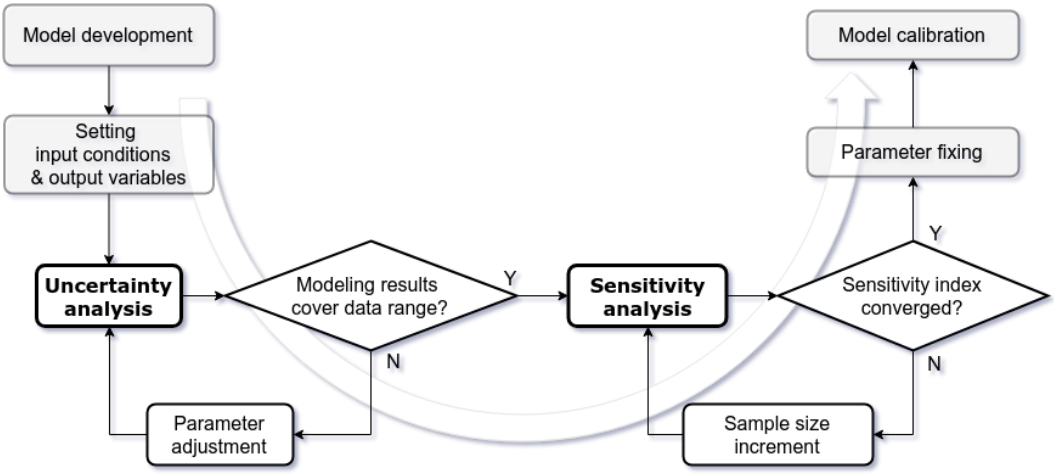

Figure 1: Workflow for performing uncertainty and sensitivity analysis.

where $SI^{ub}_{i,t}$ and $SI^{ub}_{i,t}$ are the upper and lower bound of the SI from the specific model output (i) at time (t). $CI_{i,t}$ is the corresponding convergence index.

To integrate the random phase-shift with eFAST, we built the `rfast99` function, which can be used to perform the sensitivity test for the whole modeling process. The function `rfast99` in **pksensi** can sample and generate the testing parameter matrix based on a given argument of sample size n and probability distribution (e.g., mean/s.d. of normal distribution, meanlog/sdlog of log-normal distribution, and minimum/maximum of uniform distribution). The number of model evaluations is equal to the sample size times the number of model parameters.

Modeling

The **pksensi** package provides a useful function to solve ODEs in PK models using analytical and numerical approaches. The solution can perform under the pure R programming environment by linking **pksensi** with the **deSolve** package (Soetaert et al., 2010). However, the most efficient way to achieve high computational speed is through compiled, lower-level languages, such as FORTRAN, C, or C++. The **pksensi** package includes a function to compile and create dynamic-link libraries (.dll) on Windows and shared objects (.so) on Unix-liked systems (e.g., Linux and MacOS). This compiled program can load and execute through the build-in function in R. However, Windows users need to install Rtools to compile the source code by using the GNU GCC compiler. The C implementation of the example model can be found as an example within GNU MCSim (Bois, 2009). The **pksensi** can also link with GNU MCSim to create the model program, used in solving each system of equations.

The uncertainty analysis is a crucial modeling process within SA. Sometimes, it is difficult to have informative data to set up the parameter distribution. A wider range for a parameter distribution might cause a numerical error at extreme values. On the other hand, narrower ranges might cause the incorrect calibrated result of the model parameter. The uncertainty analysis using Monte Carlo is an appropriate approach in the data-driven modeling process to address this problem. Before fitting the data to model, the Monte Carlo can be applied with the given distribution of model parameter to examine whether the experimental data are within the range of the distribution of model output. **pksensi** also has a function that can link to GNU MCSim to conduct Monte Carlo analysis, which is built-into GNU MCSim and examine the simulation result. The general workflow of uncertainty and SA is illustrated in Figure 1.

Output visualization and decision support

The output of SA returns a list that contains the given time-points and values of all state variables (e.g., chemical concentrations in blood). This format is particularly suited for further graphical routines within **pksensi**. Also, time-dependent sensitivity measurements of first and total effects for all parameters can be plotted simultaneously. If the computed SI show different values across replications under the same sample size, this indicates an inadequate sample number, which can create an unreliable result. It is essential to use a sufficient sample size to prevent incorrect judgments in parameter fixing. **pksensi** includes functions to check the convergence and sensitivity of model parameters, providing a means to assess the robustness of the sensitivity measurement. We also developed a "cut-off"-based approach in this package to distinguish between "influential" and "non-

influential" parameters. Finally, **pksensi** provides visualization tools for the effective investigation and communication of results in decision support.

Example 1: One-compartment model

In the following section, two PK models will apply to demonstrate how **pksensi** works. The first one is a generic model, and the second is a chemical-specific model.

Equations

In the first example, we use a generic, one-compartment PK model from **httk** package (Pearce et al., 2017) to demonstrate how **pksensi** can apply to PK studies. The differential equations for this model is as,

$$\frac{dA_{gutlumen}}{dt} = -k_{gutabs} \cdot A_{gutlumen} + g(t)$$
$$\frac{dA_{rest}}{dt} = k_{gutabs} \cdot A_{gutlumen} - k_{elim} \cdot A_{rest}$$

where $A_{gutlumen}$ is the state variable that describes the quantity of compound in the gut lumen (mg) and A_{rest} is the quantity of compound in rest of body and blood (mg). The parameter k_{gutabs} is the absorption rate constant that describes the chemical absorption from the gut lumen into gut tissue through first-order processes (/h) and k_{elim} is the elimination rate constant (/h), which is equal to the total clearance divided by the volume of distribution. The time-dependent function $g(t)$ is used to describe the oral dosing schedule.

The concentration of the chemical in the rest of body and blood (C_{rest}, mg/L) can calculate as,

$$C_{rest} = A_{rest}/V_{dist} \cdot BW$$

where V_{dist} is the volume of distribution (L/kg BW) and BW is the body weight (kg). C_{rest} can also represent as the chemical concentration in plasma that can be further used to compare with observed results in a PK experiment. The bioavailability is assumed to be 100% in this model.

Model implementations with deSolve package

To start, we implemented the one-compartment PK model in R. The **pksensi** allows users select the preferred method to solve the PK model, either with the **deSolve** package or with GNU MCSim through the compile function. This section mainly focuses on how to conduct global SA with **deSolve** package with pure R environment.

The one-compartment PK model can describe the quantity of compound in the gut lumen (`Agutlument`) and the rest of body (`Acompartmant`). The `Ametabolized` is the quantity of compound transform and metabolize through hepatic clearance.

```
pbtk1cpt <- function(t, state, parameters) {
  with(as.list(c(state, parameters)), {
    dAgutlument = - kgutabs * Agutlument
    dAcompartment = kgutabs * Agutlument - ke * Acompartment
    dAmetabolized = ke * Acompartment
    Ccompartment = Acompartment / vdist * BW;
    list(c(dAgutlument, dAcompartment, dAmetabolized),
         "Ccompartment" = Ccompartment)
  })
}
```

The parameter values and initial states need to be assigned to specific values before simulation. Here, we use the corresponding parameter value of acetaminophen (APAP) in this example. These model parameters are derived from the *in-vivo* or *in-vitro* experiment results. The parameter value can be generated from `parameterize_1comp` function in **httk** package as:

```
library(httk)
pars1comp <- (parameterize_1comp(chem.name = "acetaminophen"))
parms <- c(vdist = pars1comp$Vdist,
           ke = pars1comp$kelim,
```

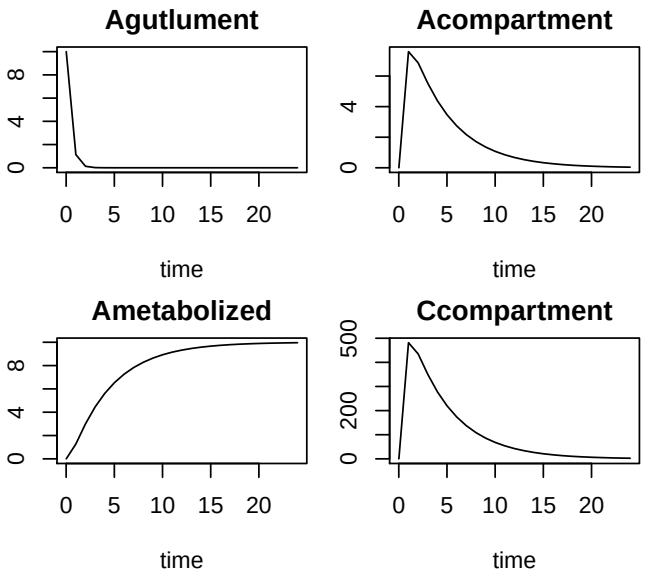

Figure 2: Simulation results of one-compartment PK model for fixed default parameters.

```
          kgutabs = pars1comp$kgutabs,
          BW = pars1comp$BW)
initState <- c(Agutlument = 10, Acompartment = 0, Ametabolized = 0)
```

The given value of `vdist`, `ke`, and `kgutabs` in **httk** are 1.1 (L/kg BW), 0.23 (/h), and 2.18 (/h), respectively. The body weight is assumed to be 70 (kg).

Here shows the given parameter value (`parms`) and initial state condition (`initState`) that need to specify in model solving. Both `parms` and `initState` are "numeric variables" that contain the value of parameter and initial state condition.

Using the `ode` function in **deSolve** package, we can visualize the PK profile (Figure 2) according to the given parameter baseline and the time points (`t`).

```
library(deSolve)
t <- seq(from = 0.01, to = 24.01, by = 1)
y <- ode(y = initState, times = t, func = pbtk1cpt, parms = parms)

par(mar=c(4,2,2,1))
plot(y)
```

We conduct global SA for the four parameters in one-compartment PK model to investigate the parameter impact on the plasma concentration during the 24-hr time post intake. The distribution of parameter is taken to be uniform with bounds corresponding to 50% and 200% of the nominal value. Therefore, the parameter ranges are assumed to be 0.55 and 2.2 L/kg BW for `vdist`. The `ke` are ranged from 0.12 to 0.46 /h, corresponding to half-times of 1.5 and 5.8 hr. The `ka` are ranged 1.09 to 4.36 /h, corresponding to half-times of 0.32 and 0.64 hr. The `BW` is assumed to a normal distribution with mean = 60 kg and sd = 5 kg.

```
q <- c("qunif", "qunif", "qunif", "qnorm")
q.arg <- list(list(min = pars1comp$Vdist / 2, max = pars1comp$Vdist * 2),
              list(min = pars1comp$kelim / 2, max = pars1comp$kelim * 2),
              list(min = pars1comp$kgutabs / 2, max = pars1comp$kgutabs * 2),
              list(mean = pars1comp$BW, sd = 5))
params <- c("vdist", "ke", "kgutabs", "BW")
```

Here we use a sample size of 200 with 10 replications. Through `rfast99` function, a S3 object with class 'rfast99' will be created. The `set.seed` can use to reproduce the same parameter matrix in the random sampling. The sample size determines the robustness of the result of SA. Higher number of sample size lead to narrower confidence intervals for sensitivity measurements across different replications. However, it will take a longer time in computation.

```
set.seed(1234)
x <- rfast99(params, n = 200, q = q, q.arg = q.arg, replicate = 10)
```

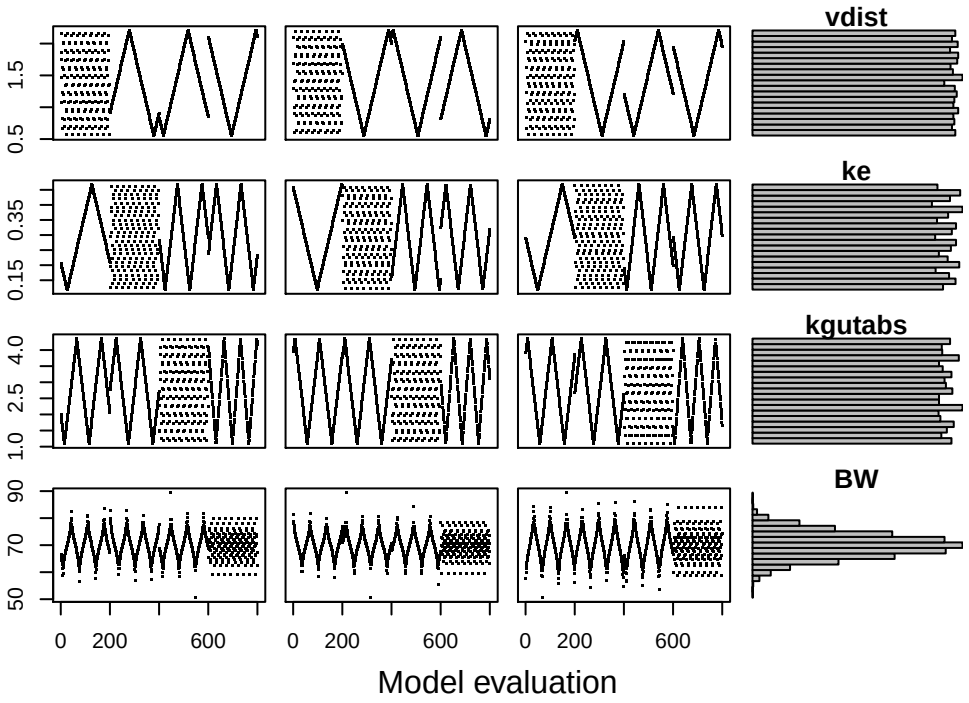

Figure 3: The parameter search curves generated by eFAST method. The x-axis is the sample number and y-axis is the parameter value for `vdist`, `ke`, `kgutabs`, and `BW` (top to down), respectively. The bar plot shows the distribution of sampling parameter.

The generated parameters are stored as a 3-D array under the named `a`, with the dimension of sample size, the number of replications, and the number of parameters, respectively.

```
dim(x$a)
```

```
  [1] 800  10   4
```

The sample number is 200, with 4 model parameters, which generates 800 model evaluations. The replication is set to 10. Therefore, the total of 8,000 parameter sequence will be used to compute the corresponding outputs. Figure 3 plotted the sampling process for each parameter from the first 3 replications. The search curves show the different intensity of sampling patterns in each segment.

```
par(mfrow=c(4,4),mar=c(0.8,0.8,0.8,0),oma=c(4,4,2,1), pch =".")
for (j in c("vdist", "ke", "kgutabs", "BW")) {
  if ( j == "BW") {
    plot(x$a[,1,j], ylab = "BW")
  } else plot(x$a[,1,j], xaxt="n", ylab = "")
  for (i in 2:3) {
  if ( j == "BW") {
    plot(x$a[,i,j], ylab = "", yaxt="n")
    } else plot(x$a[,i,j], xaxt="n", yaxt="n", ylab = "")
  }
  hist <- hist(x$a[,,j], plot=FALSE,
               breaks=seq(from=min(x$a[,,j]), to=max(x$a[,,j]), length.out=20))
  barplot(hist$density, axes=FALSE, space=0, horiz = T, main = j)
}
mtext("Model evaluation", SOUTH<-1, line=2, outer=TRUE)
```

Because the PK model is being used to describe a continuous process for the chemical concentration over time, the sensitivity measurements, therefore, have the time-dependent relationships for each model parameter. Here we use the defined output time points (`t`) to examine the change of the parameter sensitivity over time. To solve the model through **deSolve**, we need to provide the details of the argument, which include time (`t`), initial conditions of state variable (`initState`), output variables (`outnames`), and name of the model function (`func`). To create the time-dependent sensitivity measurement, we set the time duration from 0.01 to 24.01 hours with the time segment of 1 hour as the above definition in `ode` function in this example. The initial time point should avoid 0 to prevent

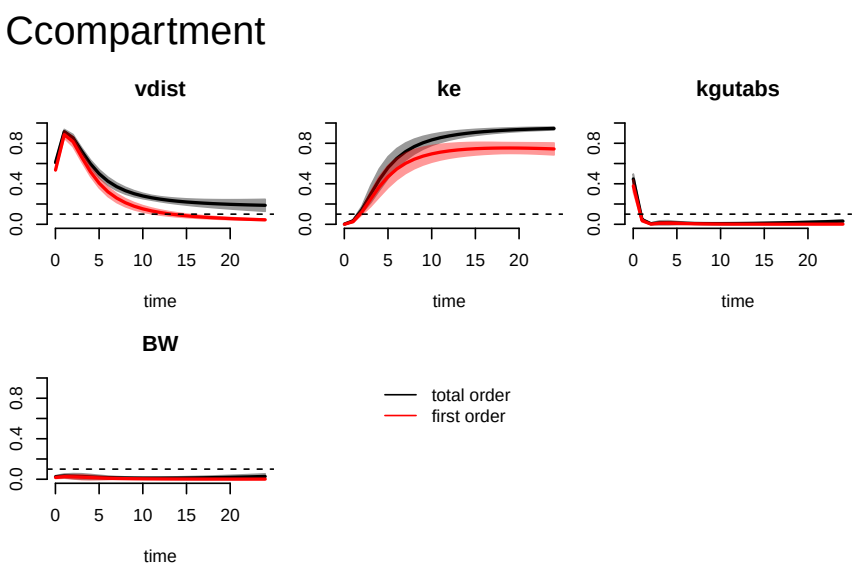

Figure 4: Time-dependent SI of the plasma concentration estimated from the one-compartment PK model during 24-hr time period intake. The solid line represents the total (black) and first (red) order SI with 95% confidence interval (polygon). The dashed line is the cut-off with the default value of 0.05.

computational error in misconduct. The `outnames` is based on the arguments from the `ode` function in **deSolve** package.

```
outputs <- c("Ccompartment", "Ametabolized")
out <- solve_fun(x, time = t, func = pbtk1cpt,
                 initState = initState, outnames = outputs)

  Starting time: 2019-10-19 11:26:22

  Ending time: 2019-10-19 11:27:53
```

The output result `out` is an S3 object of `rfast99` as well, which can link with `print`, `plot`, and `check` method to examine the sensitivity measurements. The `print` function gives the sensitivity and convergence indices for main, interaction, and total order at each time point. In addition to print out the result of SA, the more efficient way to distinguish the influence of model parameter is to visualize these indices. The time-dependent SI are shown in Figure 4. The SI has computed range from 0 (no impact) to 1 (high impact) and represents the contribution percentage of output variance under the given parameter distributions.

```
plot(out)
```

Here, we can see (Figure 4) that `vdist` and `ke` dominate the plasma concentration before and after 5-hr post chemical intake, respectively. The parameter `kgutabs` only plays a crucial role to determine the plasma concentration in the first hour. However, the current result only based on the distribution of model parameters for APAP. Given different input conditions (e.g., range of parameter uncertainty, chemical-dependent parameter value) the result can of course change (result not shown).

The default output in the plotting is setting at the first variable. To exam the time-dependent SI of other variables, such as `Ametabolized` in this case, we need to assign the variable name `vars = "Ametabolized"` in `plot` function (Figure 5).

```
plot(out, vars = "Ametabolized")
```

Figure 5 shows that the amount of metabolized is also determined by parameter `ke`. Same as `Ccompartment`, the `kgutabs` contribute about 30 - 40% variation of model output in the first hour. The `BW` is the least important parameter in the current analysis, and therefore, can be fixed in the model fitting to data and additional applications.

In addition to using the time-SI profile to investigate the parameter impact on model output, we can directly examine the relationship between parameters and model output graphically.

```
par(mfcol=c(4,4),mar=c(0.8,0.8,0,0),oma=c(4,4,2,1), pch = ".")
plot(x$a[,1,"vdist"], out$y[,1,"0.01",1], xaxt="n", main = "\nvdist")
```

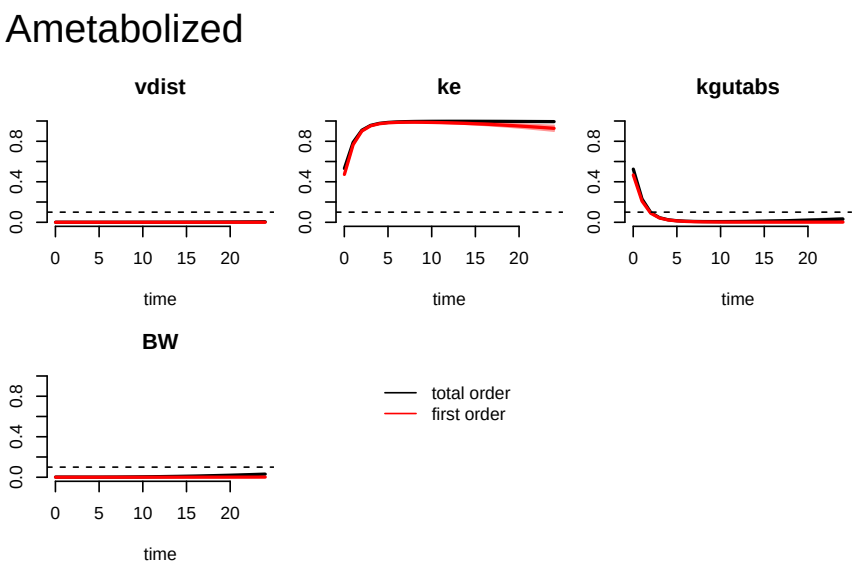

Figure 5: Time-dependent SI of the plasma concentration estimated from the one-compartment PK model during 24-hr time period intake.

```
plot(x$a[,1,"vdist"], out$y[,1,"2.01",1], xaxt="n")
plot(x$a[,1,"vdist"], out$y[,1,"6.01",1], xaxt="n")
plot(x$a[,1,"vdist"], out$y[,1,"24.01",1])
for (j in c("ke", "kgutabs", "BW")){
  for (k in c("0.01", "2.01", "6.01", "24.01")){
    if (k == "0.01") {
      plot(x$a[,1,j], out$y[,1,k,1], yaxt = "n", xaxt="n", main = paste0("\n", j))
    } else if (k == "24.01") {
      plot(x$a[,1,j], out$y[,1,k,1], yaxt = "n")
      } else plot(x$a[,1,j], out$y[,1,k,1], xaxt = "n", yaxt = "n")
  }
}
mtext("Parameter", SOUTH<-1, line=2, outer=TRUE)
mtext("Ccompartment", WEST<-2, line=2, outer=TRUE)
```

Figure 6 shows the relationship between the concentration of the rest of the body (`Ccompartment`) and the model parameters at different time points. We can find that `kgutabs` and `vdist` have higher correlation with `Ccompartment` in the beginning (t = 0.01 h) of post-intake duration compared with other parameters, suggesting that the parameters have high impact on the modeling result. The `ke` shows a high correlation at the later time period (t = 24.01 h). The parameter `BW` does not show any obvious relationship with `Ccompartment`.

The output variable `out` containing all the input arguments detailed above and the calculated SI of first order (`mSI`), interaction (`iSI`), and total order (`tSI`). Convergence indices are also stored in the list named `mCI`, `iCI`, and `tCI`. The outputs are formatted as 4-D array in `y` with the dimension name of model evaluation, number of replications, number of time points, and number of output variables, respectively.

```
dim(out$y)
```

```
  [1] 800  10  25   2
```

Some functions in **pksensi** provide efficient ways to check the result from global SA. The `check` can determine which parameters have relatively lowered sensitivity measurement across the given time points and model outputs, and therefore can be applied parameter fixing in model calibration. The `check` also provides an argument to specify the target output or change the cut-off value. The argument of `SI.cutoff` set for example at 0.05, is used to detect the relative non-influential parameters as default, in this case representing a 5% change of the output is contributed from the specific parameter variation.

```
check(out, SI.cutoff = 0.05)
```

```
  Sensitivity check ( Index > 0.05 )
```

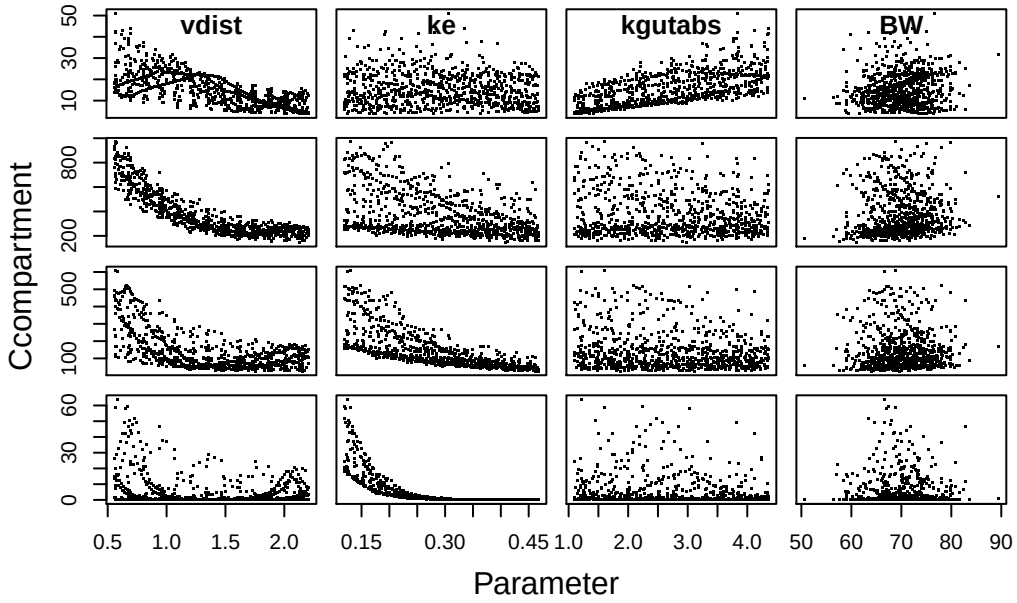

Figure 6: The relationship between model parameter and estimated concentration at times 0.01, 2.01, 6.01, and 24.01 hr (top to bottom), respectively.

```
-----------------------------------
First order:
 vdist ke kgutabs

Interaction:
 vdist ke kgutabs

Total order:
 vdist ke kgutabs

Unselected factors in total order:
 BW


Convergence check ( Index > 0.05 )
-----------------------------------
First order:
 ke kgutabs

Interaction:
 vdist

Total order:
 vdist ke
```

Based on the sensitivity measurement of the total order, the result shows that `BW` has a relative lower measurement of SI. However, all parameters do not converge to the setting cut-off, which means the larger sample size is required in further sensitivity testing. Similar to the `plot` function that can assign specific output variable in the examination, the `check` function can also use the assignment (`vars`) to examine a given output.

Model implementation with GNU MCSim

In addition to using **deSolve** to solve differential equations in PK model, the GNU MCSim can also be used, and provides better computational efficiency. To start, the users need to install GNU MCSim and related standard C compiler to solve the ODE function. The following script will download the one-compartment PK model from https://github.com/nanhung/pksensi/blob/master/tests/pbtk1cpt.model.

```
pbtk1cpt_model()
```

The computing time of using `solve_fun` in SA is estimated as,

```
system.time(out <- solve_fun(x, time = t,
                             func = pbtk1cpt, initState = initState,
                             outnames = outputs))
```

```
  Starting time: 2019-10-19 11:27:54

  Ending time: 2019-10-19 11:29:25

     user  system elapsed
   90.415   0.008  90.452
```

Then, before we conduct the SA through GNU MCSim, The following codes are used to compile the GNU MCSim model code to the executable program. To solve ODEs through GNU MCSim, we need to use `compile_model` with argument `application = mcsim`.

```
mName <- "pbtk1cpt"
compile_model(mName, application = "mcsim")
```

```
  * Created executable file 'mcsim.pbtk1cpt'.
```

Similar to `solve_fun` function that can define the initial values of parameters and state variable through generated functions, the `solve_mcsim` also has a `condition` argument that can be used to give the specific input value such as exposure dose, nominated parameter value, or initial condition of state variables.

```
conditions <- c("Agutlument = 10")
system.time(out <- solve_mcsim(x, mName = mName, params = params,
                               vars = outputs, time = t,
                               condition = conditions))
```

```
  Starting time: 2019-10-19 11:29:25

  * Created input file "sim.in".

  Execute: ./mcsim.pbtk1cpt sim.in

  Ending time: 2019-10-19 11:29:27

     user  system elapsed
    1.772   0.041   1.801
```

After solving the equations under the same given condition, we can find that GNU MCSim has more than 10 times faster in computing performance than using **deSolve**. In this case, we only focus on performing the global SA alone for generic PK model without additional comparison with experimental data. The next example will display and reproduce our previous published result (Hsieh et al., 2018) with full global SA workflow.

Example 2: Acetaminophen-PBPK model

This section aims to apply global SA for APAP-PBPK model through **pksensi**. The model describes the PK of parent APAP and its two metabolites glucuronide (APAP-G) and sulfate (APAP-S). The model code is included in this package in `https://github.com/nanhung/pksensi/blob/master/tests/pbpk_apap.model` and can generate through `pbpk_apap_model` function. We apply the global SA workflow to the original published model with 21 model parameters (Zurlinden and Reisfeld, 2016). The descriptions of each parameter and the sampling range are list in Table 1.

As in the example of the one-compartment PK model, the model parameter and the corresponding sampling range should be defined to create the parameter matrix. Previously, the probability distributions of model parameters were set to either truncated normal or uniform distributions when the parameters have informative prior information or not, respectively. To rapidly reach the acceptable convergence in this example, we use a uniform distribution for all testing parameters. The ranges of informative parameters set to 1.96-times difference for the single side under log-scaled (approximate 54.6 times difference between minimum and maximum in natural scaled). The nominal values of informative model parameters define as:

Parameter	Description	Unit	Min	Max
Tg	Gatric emptying time constant	h	-3.43	0.49
Tp	GI perfusion time constant	h	-5.37	-1.45
CYP_Km	Cytochrome P450 metabolism, Km	μM	3.00	7.00
CYP_VmaxC	Cytochrome P450 metabolism, VMax	$\mu mole/h \cdot BW^{0.75}$	-1.97	7.97
SULT_Km_apap	Sulfation pathway acetaminophen, Km	μM	3.74	7.66
SULT_Ki	Sulfation pathway substrate inhibition, Ki	μM	4.30	8.22
SULT_Km_paps	Sulfation pathway PAPS, Km	$-$	-3.00	1.00
SULT_VmaxC	Sulfation pathway acetaminophen, Vmax	$\mu mole/h \cdot BW^{0.75}$	0.00	10.00
UGT_Km	Glucuronidation pathway acetaminophen, Km	μM	6.74	10.66
UGT_Ki	Glucuronidation pathway substrate inhibition, Ki	μM	9.01	12.93
UGT_Km_GA	Glucuronidation pathway GA, Km	$-$	-3.00	1.00
UGT_VmaxC	Glucuronidation pathway acetaminophen, Vmax	$\mu mole/h \cdot BW^{0.75}$	0.00	10.00
Km_AG	APAP-G hepatic transporter, Km	μM	7.94	11.86
Vmax_AG	APAP-G hepatic transporter, Vmax	$\mu mole/h$	6.99	15.00
Km_AS	APAP-S hepatic transporter, Km	μM	8.00	12.00
Vmax_AS	APAP-S hepatic transporter, Vmax	$\mu mole/h$	6.99	15.00
kGA_syn	UDPGA synthesis	$1/h$	0.00	13.00
PAPS_syn	PAPS synthesis	$1/h$	0.00	13.00
CLC_APAP	APAP clearance	$L/h \cdot BW^{0.75}$	-6.00	1.00
CLC_AG	APAP-G clearance	$L/h \cdot BW^{0.75}$	-6.00	1.00
CLC_AS	APAP-S clearance	$L/h \cdot BW^{0.75}$	-6.00	1.00

Table 1: Description of sampling range of model parameter.

```
Tg <- log(0.23)
Tp <- log(0.033)
CYP_Km <- log(130)
SULT_Km_apap <- log(300)
SULT_Ki <- log(526)
SULT_Km_paps <- log(0.5)
UGT_Km <- log(6.0e3)
UGT_Ki <- log(5.8e4)
UGT_Km_GA <-log(0.5)
Km_AG <- log(1.99e4)
Km_AS <- log(2.29e4)
rng <- 1.96
```

Generally, a wide range of parameter value might cause the computing error when solving the differential equation. One of the effective ways to prevent this problem is to adjust the value of relative and absolute error tolerance to control the error appearance by resetting these parameters in the lower levels. The `generate_infile` and `solve_mcsim` functions provide arguments of `rtol` and `atol` that adjust the error tolerance to prevent the unwanted error. However, the modification will decrease the computing speed. Therefore, the alternative method to prevent this issue is to detect the crucial parameter range that causes the problem. Also, setting the maximum number of steps to the higher value in GNU MCSim can prevent this problem (internally defined). The maximum number of step is 5000 (default) in this case. The value can reset through reinstallation of GNU MCSim. Here we separate the global SA of APAP-PBPK model process to several steps.

Prepare and compile the model file

The model code needs to prepare in the following global SA workflow. After creating the `pbpk_apap.model` file in the working directory, the next step is to generate the executable program (`mcsim.pbpk_apap`) through the `compile_model` function.

```
mName <- "pbpk_apap"
pbpk_apap_model()
compile_model(mName, application = "mcsim")

  * Created executable file 'mcsim.pbpk_apap'.
```

Define the parameter and its distribution

The 21 tested model parameters are defined in this step, including parameter names and probability distributions. To prevent errors, the ranges of `SULT_VmaxC` and `UGT_VmaxC` need to be adjusted from $U(0, 15)$ (Zurlinden and Reisfeld, 2016) to $U(0, 10)$ (Hsieh et al., 2018). The objects `q` and `dist` are set

to the type of distribution that will be used to generate the parameter matrix in GNU MCSim (for uncertainty analysis) and R (for SA).

```
params <- c("lnTg", "lnTp", "lnCYP_Km","lnCYP_VmaxC",
           "lnSULT_Km_apap","lnSULT_Ki","lnSULT_Km_paps","lnSULT_VmaxC",
           "lnUGT_Km","lnUGT_Ki","lnUGT_Km_GA","lnUGT_VmaxC",
           "lnKm_AG","lnVmax_AG","lnKm_AS","lnVmax_AS",
           "lnkGA_syn","lnkPAPS_syn", "lnCLC_APAP","lnCLC_AG","lnCLC_AS")
dist <- rep("Uniform", 21)
q <- rep("qunif", 21)
q.arg <-list(list(Tg-rng, Tg+rng), list(Tp-rng, Tp+rng),
             list(CYP_Km-rng, CYP_Km+rng), list(-2., 5.),
             list(SULT_Km_apap-rng, SULT_Km_apap+rng),
             list(SULT_Ki-rng, SULT_Ki+rng),
             list(SULT_Km_paps-rng, SULT_Km_paps+rng),
             list(0, 10), list(UGT_Km-rng, UGT_Km+rng),
             list(UGT_Ki-rng, UGT_Ki+rng),
             list(UGT_Km_GA-rng, UGT_Km_GA+rng),
             list(0, 10), list(Km_AG-rng, Km_AG+rng),
             list(7., 15), list(Km_AS-rng, Km_AS+rng),
             list(7., 15), list(0., 13), list(0., 13),
             list(-6., 1), list(-6., 1), list(-6., 1))
```

Define additional input condition and output time and variables

In this case, we only use GNU MCSim to estimate the concentration of APAP and its metabolites in plasma to optimize the computing speed. The oral dose of APAP is 20 mg/kg in this example. Generally, the input dosing method can be defined through the `condition` argument. Since the unit of the given dose is mg/kg, the `mgkg_flag` is set to 1. Additional input schedule functions can be found in the section of input functions in GNU MCSim User's Manual (https://www.gnu.org/software/mcsim/mcsim.html#Input-functions).

```
conditions <- c("mgkg_flag = 1",
                "OralExp_APAP = NDoses(2, 1, 0, 0, 0.001)",
                "OralDose_APAP_mgkg = 20.0")
vars <- c("lnCPL_APAP_mcgL", "lnCPL_AG_mcgL", "lnCPL_AS_mcgL")
times <- seq(from = 0.1, to = 12.1, by = 0.2)
```

Uncertainty analysis

We apply uncertainty analysis through the `solve_mcsim` function and visualize the results using `pksim` function. Some example data are included in **pksensi** with experiment time (h) and concentration (mg/L).

```
head(APAP)
```

```
    Wt Time APAP    AG   AS         Study Dose
  1 72 0.25 15.5  3.20 3.55 Prescott_1980   20
  2 72 0.50 18.7  7.95 6.75 Prescott_1980   20
  3 72 0.75 18.8 13.42 8.21 Prescott_1980   20
  4 72 1.00 18.0 18.14 9.56 Prescott_1980   20
  5 72 1.50 15.6 24.68 9.35 Prescott_1980   20
  6 72 2.00 13.6 27.28 8.78 Prescott_1980   20
```

In setting simulation conditions, the relative and absolute error tolerance (`rtol` & `atol`) were set to 1e-7 and 1e-9, respectively, to prevent the numerical errors. The Monte Carlo simulation is run for 1000 iterations in the assignment of `monte_carlo`. The input file (`sim.in`) and output file (`simmc.out`) will be generated under the standard ASCII format.

```
set.seed(1111)
out <- solve_mcsim(mName = mName, params = params, vars = vars,
                   monte_carlo = 1000, dist = dist, q.arg = q.arg,
                   time = times, condition = conditions,
                   rtol = 1e-7, atol = 1e-9)
```

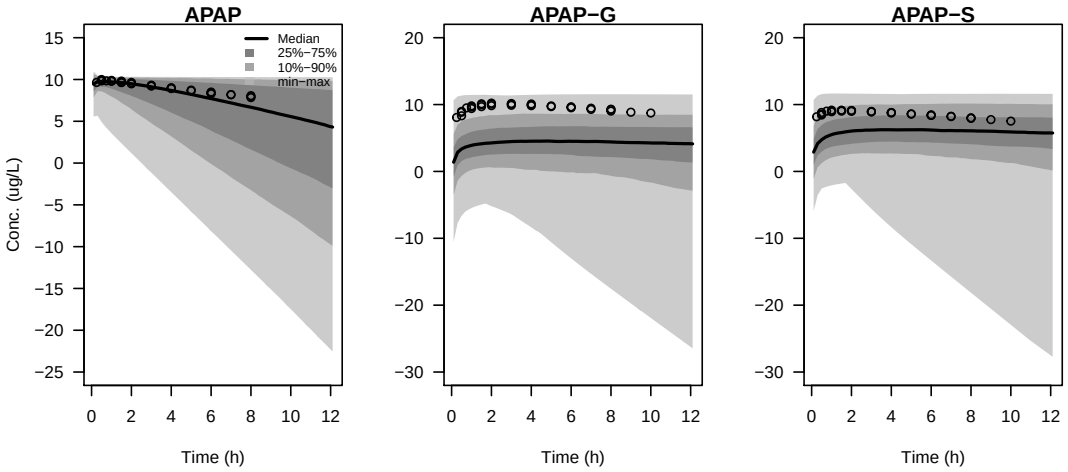

Figure 7: Coverage checks of prior PBPK model predictions with calibrated APAP data.

```
Starting time: 2019-10-19 11:29:28

* Created input file "sim.in".

Execute: ./mcsim.pbpk_apap sim.in

Ending time: 2019-10-19 11:29:30
```

Figure 7 shows the uncertainty analysis of APAP data and modeling results. For parent compound, all data points are located in the simulated interval of 25-75%, suggesting that the simulated outputs can accurately cover the experimental concentration profiles under the assumed parameter ranges for APAP. The simulated results of metabolites APAP-G and APAP-S, while not fully in the 25-75% range, and nonetheless within the envelope of the simulated intervals.

```
par(mfrow = c(1,3), mar = c(4,4,1,1))
pksim(out, xlab = "Time (h)", ylab = "Conc. (ug/L)", main = "APAP")
points(APAP$Time, log(APAP$APAP * 1000))
pksim(out, vars = "lnCPL_AG_mcgL", xlab = "Time (h)", main = "APAP-G",
      ylab = " ", legend = FALSE)
points(APAP$Time, log(APAP$AG * 1000))
pksim(out, vars = "lnCPL_AS_mcgL", xlab = "Time (h)", main = "APAP-S",
      ylab = " ", legend = FALSE)
points(APAP$Time, log(APAP$AS * 1000))
```

Generate parameter matrix

To perform global SA, we next have to generate the parameter matrix from the eFAST method. An example with sample size 512 with 10 replications would be as follows:

```
set.seed(1234)
x <- rfast99(params = params, n = 512, q = q, q.arg = q.arg, replicate = 10)
```

Conduct the global SA

To conduct the global SA with GNU MCSim and **pksensi**, an input file needs to be generate before modeling. The file can be created by the `generate_infile` function. Alternatively, `solve_mcsim` can also automatically create the input file and compute the output.

```
out <- solve_mcsim(x, mName = mName,
                   params = params,
                   time = times,
                   vars = vars,
                   condition = conditions,
                   rtol = 1e-7, atol = 1e-9)
```

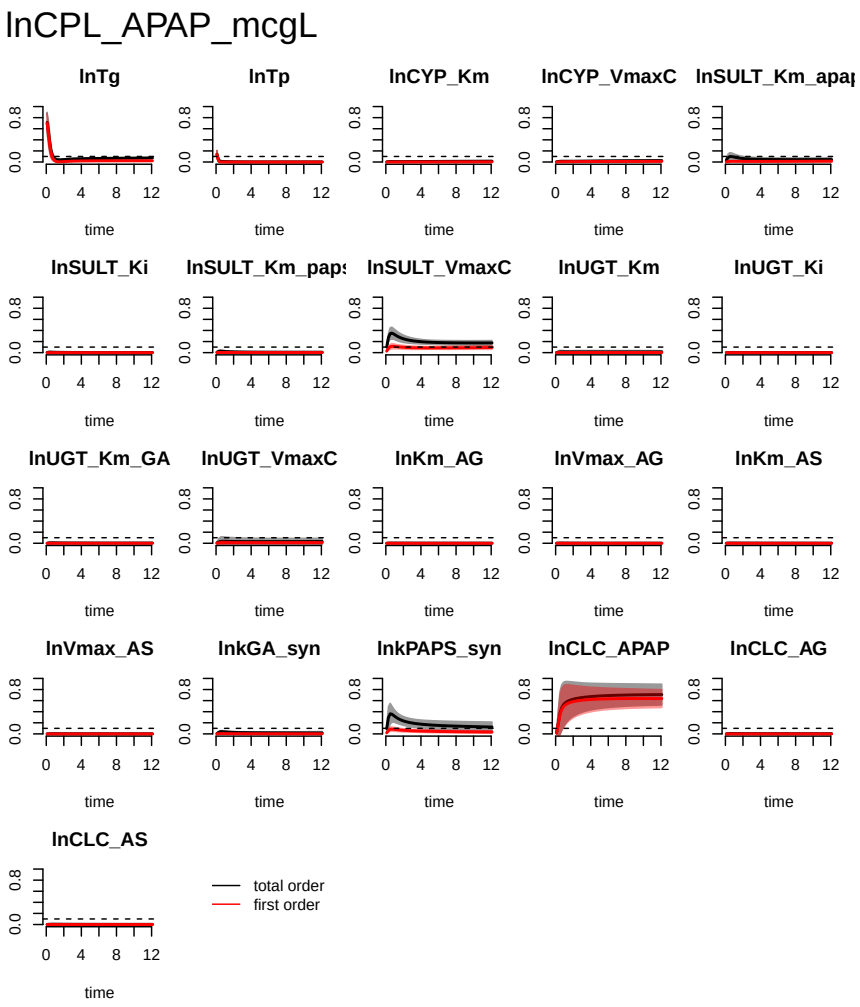

Figure 8: Time-dependent SI of the plasma APAP concentration estimated from APAP-PBPK model during 12-hr time period post intake.

```
Starting time: 2019-10-19 11:29:31

* Created input file "sim.in".

Execute: ./mcsim.pbpk_apap sim.in

Ending time: 2019-10-19 11:33:25
```

Visualization and decision

The plotting function can show the results of time-dependent sensitivity to determine the parameter impacts on model output over time (Figure 8).

```
plot(out, vars = "lnCPL_APAP_mcgL")
```

Based on our previous study, we propose the heatmap visualization approach `heat_check` to distinguish "influential" and "non-influential" parameters with a "cut-off" point. Through the given argument `order`, we can select the specific order of sensitivity measurement that we are interested in (Figure 9).

```
heat_check(out, order = "total order")
```

Adding the `index = "CI"` in the function can further investigate the convergence index. Based on the current setting of sampling size, most parameters cannot reach the acceptable criteria of convergence (Figure 10). Therefore, a higher number of sampling is necessary. The sample size of convergence in the current PBPK model is 8,192 (Hsieh et al., 2018). However, based on the current sample size, we still can find 6 parameters that can be an important parameter for the plasma APAP concentration.

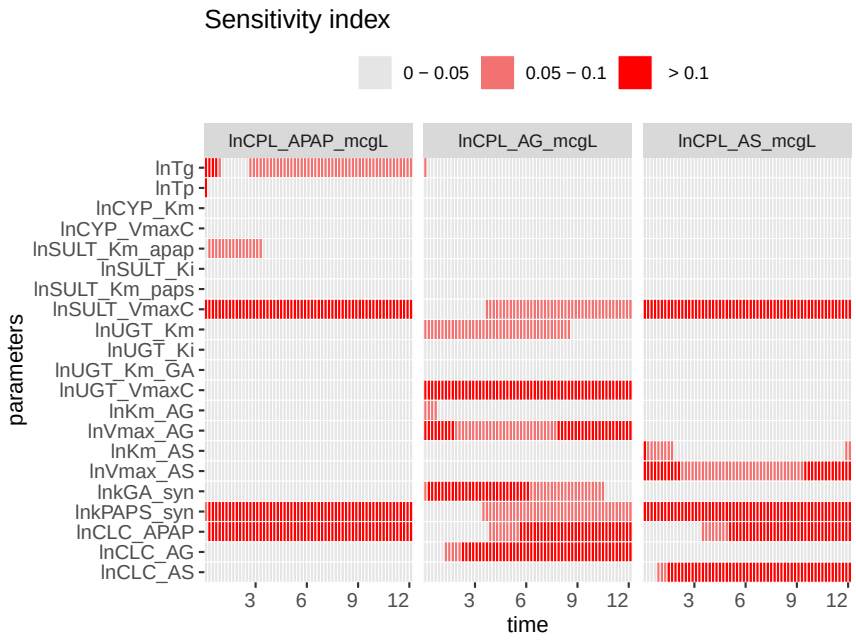

Figure 9: Heat map representation of time-dependent total SI computed in global SA method.

```
heat_check(out, index = "CI", order = "total order")
```

Conclusions and perspectives

We initiated **pksensi** in order to facilitate a comprehensive and efficient global SA workflow in PK modeling, filling a crucial gap for open-source modeling communities. It provides a straightforward application to investigate the impact of parameters on model outputs across simulation time-points and variables. We adopted the eFAST method, a common variance-based SA that we found to have the best balance of efficiency and accuracy. In addition, the package also includes the ability to assess the convergence of SA results, which is rarely addressed in most global SA studies and software packages. We also developed functions to visualize the output results and help distinguish the "influential" and "non-influential" parameters that can be applied in parameter fixing in the model calibration. In future versions, we expect to add additional global SA approaches (Glen and Isaacs, 2012; Ring et al., 2017). Other features, such as principle component analysis (Lamboni et al., 2011) will also be considered.

We chose to integrate sensitivity analysis with GNU MCSim because it is a powerful open-source software package for dynamical simulation of biological-based models, mainly built for PK research with probabilistic approaches (Bois, 2009). In addition, the current version of **pksensi** includes both GNU MCSim and **deSolve** as the main ODE solvers. Compared with the **deSolve** package, the GNU MCSim can provide better speed and efficiency to solve the ODEs in PK models. Although **pksensi** provides the essential features and functions to install and link with GNU MCSim, conducting the uncertainty and SA in PK modeling, the overall learning curve of this workflow might steep for users that are not familiar with GNU MCSim from model development, debugging, and testing. Therefore, the general package, such as **deSolve**, can be an alternative option for R users. Some other packages are also available to solve ODEs, such as **RxODE** (Wang et al., 2016) and **mrgsolve** (Baron, 2019). We hope to incorporate these packages to expand the flexibility of using **pksensi** in the future. It addition, **pksensi** also has potential to be performed in interactive PBPK modeling platform (Li et al., 2019).

It is worth noting that the current global SA results are conditional on the specific model and the given range of the model parameters. Other factors, such as the number of parameters examined, the parameter distributions/ranges, and the given dosing level in PK model might also have a potential impact on the level of importance of each parameter, but these "meta-uncertainties" are not currently addressed. Complex models with high paremeter dimensionality are also challenging in global SA as they have higher computer demands. As approaches are developed to address these complexities, we hope to incorporate them into **pksensi**.

Overall, **pksensi** provides comprehensive suite of features and functions for performing global SA for PK models and can create robust and reproducible results for decision making in model development and calibration. Although **pksensi** used here is mainly for PK modeling, it can also be applied to other ODE-based dynamic models in order to investigate the sensitivity of model outputs

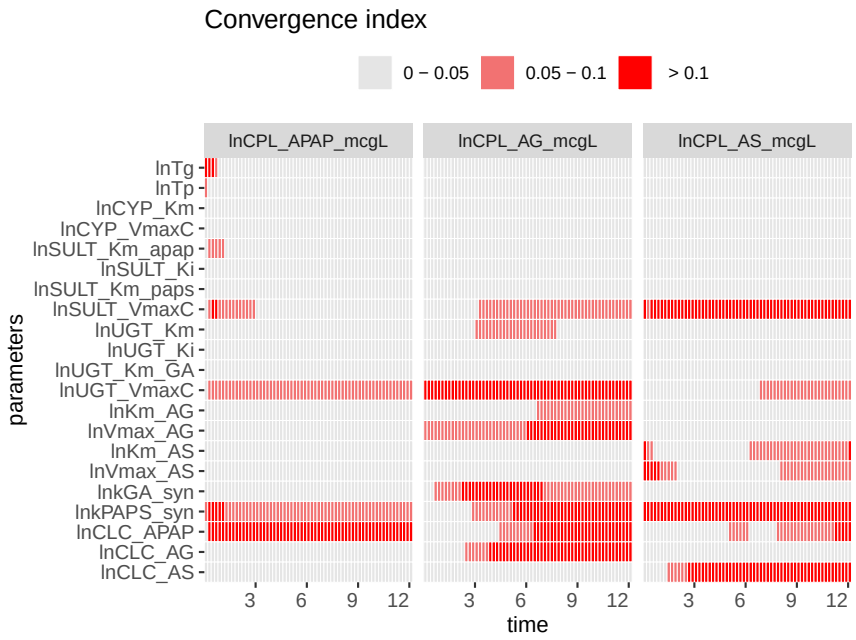

Figure 10: Heat map representation of time-dependent total CI computed in global SA method.

to input parameters.

Code

The source material for this paper is available at https://github.com/nanhung/pksensi-RJ.

Acknowledgments

This work was funded, in part, by grant 1U01FD005838 from the U.S. Food and Drug Administration (FDA) and grant P42 ES027704 from the U.S. National Institute of Environmental Health Sciences. This article reflects the views of the author and should not be construed to represent FDA's views or policies. We thank Dr. Yasha Hartberg and Dr. Barbara Gastel at the Texas A&M University, Dr. Frederic Bois from Certara and Dr. Eleftheria Tsakalozou from U.S. FDA for reviewing the manuscript and consultation.

Session information

- R version 3.6.1 (2019-07-05), `x86_64-pc-linux-gnu`
- Locale: `LC_CTYPE=en_US.UTF-8`, `LC_NUMERIC=C`, `LC_TIME=en_US.UTF-8`, `LC_COLLATE=en_US.UTF-8`, `LC_MONETARY=en_US.UTF-8`, `LC_MESSAGES=en_US.UTF-8`, `LC_PAPER=en_US.UTF-8`, `LC_NAME=C`, `LC_ADDRESS=C`, `LC_TELEPHONE=C`, `LC_MEASUREMENT=en_US.UTF-8`, `LC_IDENTIFICATION=C`
- Running under: `elementary OS 5.0 Juno`
- Matrix products: default
- BLAS: `/usr/lib/x86_64-linux-gnu/openblas/libblas.so.3`
- LAPACK: `/usr/lib/x86_64-linux-gnu/libopenblasp-r0.2.20.so`
- Base packages: base, datasets, graphics, grDevices, methods, stats, utils
- Other packages: deSolve 1.24, httk 1.10.0, pksensi 1.1.3.9000, xtable 1.8-4
- Loaded via a namespace (and not attached): assertthat 0.2.1, colorspace 1.4-1, compiler 3.6.1, crayon 1.3.4, data.table 1.12.2, DBI 1.0.0, digest 0.6.20, dplyr 0.8.3, evaluate 0.14, expm 0.999-4, getPass 0.2-2, ggplot2 3.2.1, glue 1.3.1, grid 3.6.1, gtable 0.3.0, htmltools 0.3.6, knitr 1.25, labeling 0.3, lattice 0.20-38, lazyeval 0.2.2, magrittr 1.5, Matrix 1.2-17, mitools 2.4, msm 1.6.7, munsell 0.5.0, mvtnorm 1.0-11, pillar 1.4.2, pkgconfig 2.0.2, plyr 1.8.4, purrr 0.3.2, R6 2.4.0, Rcpp 1.0.2, reshape 0.8.8, reshape2 1.4.3, rlang 0.4.0, rmarkdown 1.15, rticles 0.10, scales 1.0.0, splines 3.6.1, stringi 1.4.3, stringr 1.4.0, survey 3.36, survival 2.44-1.1, tibble 2.1.3, tidyselect 0.2.5, tools 3.6.1, xfun 0.9, yaml 2.2.0

Bibliography

K. T. Baron. *mrgsolve: Simulate from ODE-Based Models*, 2019. URL https://CRAN.R-project.org/package=mrgsolve. R package version 0.9.1. [p15]

C. Bidot, M. Lamboni, and H. Monod. *multisensi: Multivariate Sensitivity Analysis*, 2018. URL https://CRAN.R-project.org/package=multisensi. R package version 2.1-1. [p2]

F. Y. Bois. GNU MCSim: Bayesian statistical inference for SBML-coded systems biology models. *Bioinformatics*, 25(11):1453–1454, 2009. URL http://doi.org/10.1093/bioinformatics/btp162. [p1, 3, 15]

Y. Chu and J. Hahn. Quantitative optimal experimental design using global sensitivity analysis via quasi-linearization. *Industrial & Engineering Chemistry Research*, 49(17):7782–7794, 2010. URL https://doi.org/10.1021/ie9009827. [p1]

R. Cukier, C. Fortuin, K. E. Shuler, A. Petschek, and J. Schaibly. Study of the sensitivity of coupled reaction systems to uncertainties in rate coefficients. i theory. *The Journal of Chemical Physics*, 59(8): 3873–3878, 1973. URL https://doi.org/10.1063/1.1680571. [p1]

F. Ferretti, A. Saltelli, and S. Tarantola. Trends in sensitivity analysis practice in the last decade. *Science of the Total Environment*, 568:666–670, 2016. URL https://doi.org/10.1016/j.scitotenv.2016.02.133. [p1]

R. I. Garcia, J. G. Ibrahim, J. F. Wambaugh, E. M. Kenyon, and R. W. Setzer. Identifiability of PBPK models with applications to dimethylarsinic acid exposure. *Journal of Pharmacokinetics and Pharmacodynamics*, 42(6):591–609, 2015. URL https://doi.org/10.1007/s10928-015-9424-2. [p1]

G. Glen and K. Isaacs. Estimating sobol sensitivity indices using correlations. *Environmental Modelling & Software*, 37:157–166, 2012. URL https://doi.org/10.1016/j.envsoft.2012.03.014. [p15]

N.-H. Hsieh, B. Reisfeld, F. Y. Bois, and W. A. Chiu. Applying a global sensitivity analysis workflow to improve the computational efficiencies in physiologically-based pharmacokinetic modeling. *Frontiers in Pharmacology*, 9:588, 2018. URL https://doi.org/10.3389/fphar.2018.00588. [p1, 10, 11, 14]

B. Iooss, A. Janon, G. Pujol, with contributions from Khalid Boumhaout, S. D. Veiga, T. Delage, J. Fruth, L. Gilquin, J. Guillaume, L. Le Gratiet, P. Lemaitre, B. L. Nelson, F. Monari, R. Oomen, B. Ramos, O. Roustant, E. Song, J. Staum, R. Sueur, T. Touati, and F. Weber. *sensitivity: Global Sensitivity Analysis of Model Outputs*, 2018. URL https://CRAN.R-project.org/package=sensitivity. R package version 1.15.1. [p1, 2]

M. J. Jansen. Analysis of variance designs for model output. *Computer Physics Communications*, 117 (1-2):35–43, 1999. URL https://doi.org/10.1016/S0010-4655(98)00154-4. [p1]

M. Lamboni, H. Monod, and D. Makowski. Multivariate sensitivity analysis to measure global contribution of input factors in dynamic models. *Reliability Engineering & System Safety*, 96(4): 450–459, 2011. URL https://doi.org/10.1016/j.ress.2010.12.002. [p15]

M. Li, Y.-H. Cheng, J. T. Chittenden, R. E. Baynes, L. A. Tell, J. L. Davis, T. W. Vickroy, J. E. Riviere, and Z. Lin. Integration of food animal residue avoidance databank (farad) empirical methods for drug withdrawal interval determination with a mechanistic population-based interactive physiologically based pharmacokinetic (ipbpk) modeling platform: example for flunixin meglumine administration. *Archives of toxicology*, pages 1–16, 2019. URL https://doi.org/10.1007/s00204-019-02464-z. [p15]

G. D. Loizou, K. McNally, K. Jones, and J. Cocker. The application of global sensitivity analysis in the development of a physiologically based pharmacokinetic model for m-xylene and ethanol co-exposure in humans. *Frontiers in Pharmacology*, 6:135, 2015. URL https://doi.org/10.3389/fphar.2015.00135. [p1]

K. McNally, R. Cotton, and G. D. Loizou. A Workflow for Global Sensitivity Analysis of PBPK Models. *Frontiers in Pharmacology*, 2:31, 2011. URL http://doi.org/10.3389/fphar.2011.00031. [p1]

K. McNally, R. Cotton, J. Cocker, K. Jones, M. Bartels, D. Rick, P. Price, and G. Loizou. Reconstruction of exposure to m-Xylene from human biomonitoring data using PBPK modelling, Bayesian inference, and Markov Chain Monte Carlo simulation. *Journal of Toxicology*, 2012, 2012. URL http://doi.org/10.1155/2012/760281. [p1]

A. B. Owen. Better estimation of small sobol'sensitivity indices. *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, 23(2):11, 2013. URL http://doi.org/10.1145/2457459.2457460. [p1]

R. Pearce, R. W. Setzer, C. Strope, N. Sipes, and J. Wambaugh. httk: R package for high-throughput toxicokinetics. *Journal of Statistical Software*, 79(4):1–26, 2017. URL https://doi.org/10.18637/jss.v079.i04. [p4]

D. Reusser. *fast: Implementation of the Fourier Amplitude Sensitivity Test (FAST)*, 2015. URL https://CRAN.R-project.org/package=fast. R package version 0.64. [p2]

C. L. Ring, R. G. Pearce, R. W. Setzer, B. A. Wetmore, and J. F. Wambaugh. Identifying populations sensitive to environmental chemicals by simulating toxicokinetic variability. *Environment International*, 106:105–118, 2017. URL http://doi.org/10.1016/j.envint.2017.06.004. [p15]

A. Saltelli, S. Tarantola, and K.-S. Chan. A quantitative model-independent method for global sensitivity analysis of model output. *Technometrics*, 41(1):39–56, 1999. URL http://doi.org/10.2307/1270993. [p1, 2]

F. Sarrazin, F. Pianosi, and T. Wagener. Global sensitivity analysis of environmental models: convergence and validation. *Environmental Modelling & Software*, 79:135–152, 2016. URL https://doi.org/10.1016/j.envsoft.2016.02.005. [p2]

K. Soetaert and T. Petzoldt. Inverse Modelling, Sensitivity and Monte Carlo Analysis in R Using Package FME. *Journal of Statistical Software*, 33(3):1–28, 2010. URL https://doi.org/10.18637/jss.v033.i03. [p1]

K. Soetaert, T. Petzoldt, and R. W. Setzer. Solving Differential Equations in R: Package deSolve. *Journal of Statistical Software*, 33(9):1–25, 2010. URL https://doi.org/10.18637/jss.v033.i09. [p3]

J. Wang, R. Faivre, H. Richard, and H. Monod. mtk: A General-Purpose and Extensible R Environment for Uncertainty and Sensitivity Analyses of Numerical Experiments. *The R Journal*, 7(2):206–226, 2015. URL https://doi.org/10.32614/RJ-2015-031. [p2]

W. Wang, K. Hallow, and D. James. A tutorial on RxODE: simulating differential equation pharmacometric models in R. *CPT: Pharmacometrics & Systems Pharmacology*, 5(1):3–10, 2016. URL https://doi.org/10.1002/psp4.12052. [p15]

F. Weber and S. Theers. *ODEsensitivity: Sensitivity Analysis of Ordinary Differential Equations*, 2019. URL https://CRAN.R-project.org/package=ODEsensitivity. R package version 1.1.2. [p2]

X.-Y. Zhang, M. Trame, L. Lesko, and S. Schmidt. Sobol sensitivity analysis: a tool to guide the development and evaluation of systems pharmacology models. *CPT: Pharmacometrics & Systems Pharmacology*, 4(2):69–79, 2015. URL https://doi.org/10.1002/psp4.6. [p1]

T. J. Zurlinden and B. Reisfeld. Physiologically based modeling of the pharmacokinetics of acetaminophen and its major metabolites in humans using a bayesian population approach. *European Journal of Drug Metabolism and Pharmacokinetics*, 41:267–280, 2016. URL https://doi.org/10.1007/s13318-015-0253-x. [p10, 11]

Nan-Hung Hsieh
Veterinary Integrative Biosciences, College of Veterinary Medicine and Biomedical Sciences, Texas A&M University
College Station, TX, USA
nhsieh@cvm.tamu.edu

Brad Reisfeld
Chemical and Biological Engineering and School of Biomedical Engineering, Colorado State University
Fort Collins, CO, USA
brad.reisfeld@colostate.edu

Weihsueh A. Chiu
Veterinary Integrative Biosciences, College of Veterinary Medicine and Biomedical Sciences, Texas A&M University
College Station, TX, USA
wchiu@cvm.tamu.edu