Developing Shiny Applications to Simulate Pharmacometric Models:

Things I wish I knew yesterday

Jessica Wojciechowski
Email: jessica.wojciechowski@mymail.unisa.edu.au

University of South Australia
Australian Centre for Pharmacometrics

Overview

- Helpful tips in context of:
  - Simulating (and presenting the results of) pharmacometric models coded in R using differential equations
  - Controlling application speed using features from Shiny
  - Increasing application speed from R programming
  - Designing user-interfaces for others who may use your application

R and Shiny for Simulating Population Models

- Flexible R programming language can accommodate simulations of all model types and between-subject variability
- User time required to generate model predictions can be more efficient than other software
    - Graphical output produced simultaneously with changing input
- Application custom-made for the model and its purpose

Example Shiny Application

University of South Australia | Australian Centre for Pharmacometrics

Ibuprofen Dosing Regimens in Pre-Term Neonates for Patent Ductus Arteriosus

Reference: Gregoire N, Desfrere L, Roze JC, Kibleur Y, Koehne P. Population pharmacokinetic analysis of Ibuprofen enantiomers in preterm newborn infants. Journal of clinical pharmacology. 2008;48(12):1460-8.

Click the Update Plot button once dosing regimen selections below have been finalised

Update Plot

Dosing Regimen:

- (•) Loading-Bolus-Bolus
- ( ) Loading-Continuous Infusion

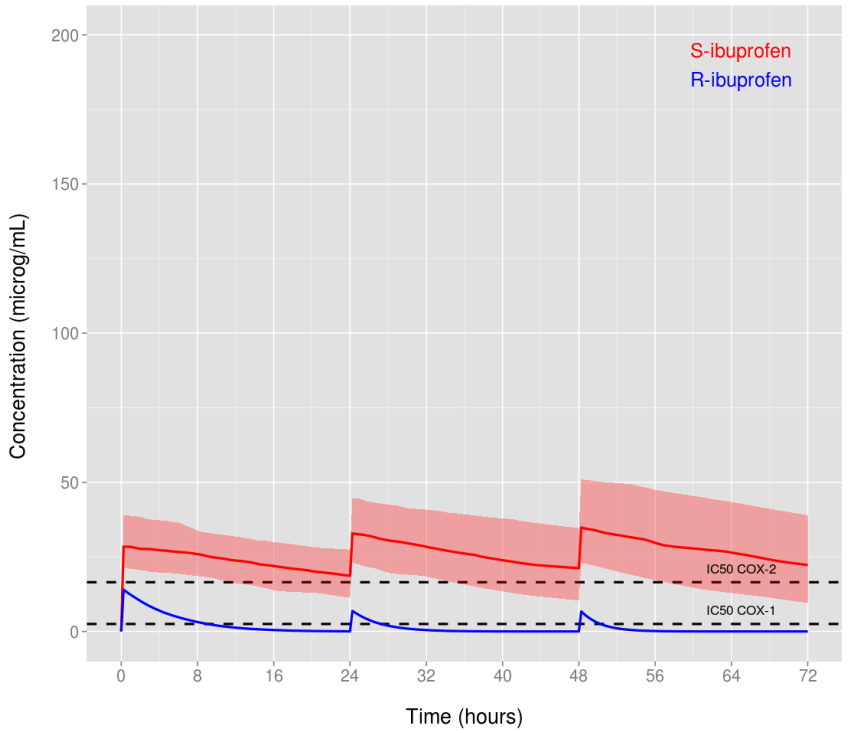
Bolus Dose at 24 and 48 hours (mg/kg): 0 | 5 | 40

Loading Dose (mg/kg): 0 | 10 | 40

Postnatal Age (Hours): 0 | 12 | 72

Number of Individuals: 10 | 100

Prediction Intervals %: 80% - 10th and 90th percentiles



Available at: https://acp-unisa.shinyapps.io/IbuprofenNeonates/

Tip 1: Start simple, start with just R

- Unless you specifically call an object to the Shiny user-interface or have it printed in the R console, you can't see it in a running Shiny application
    - i.e., time sequences, data frames, functions
- Error messages resulting from these appear in the R console
    - But you can't call the objects to check what's going on

Tip 1: Start simple, start with just R

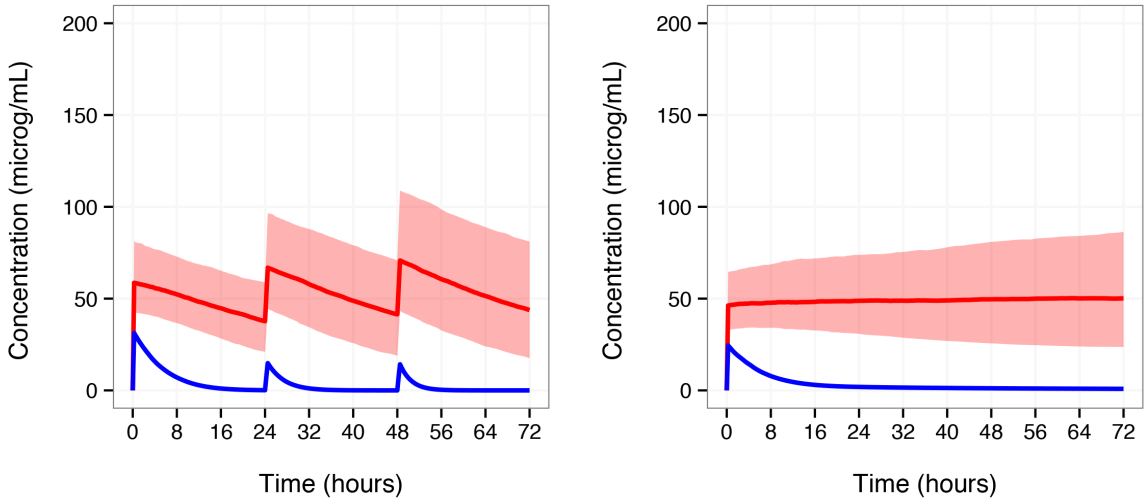
- Code your model in a standard R script first
  - Build your script sequentially and test what objects look like (and if they look right) before going on to the next step
    - Test what your potential output will look like
  - Allows you to address the “non-Shiny related” error messages first
  - Requires manual changing of “user-defined input” in the code, such as dose and covariate values

Tip 2: Perform Independent Checks

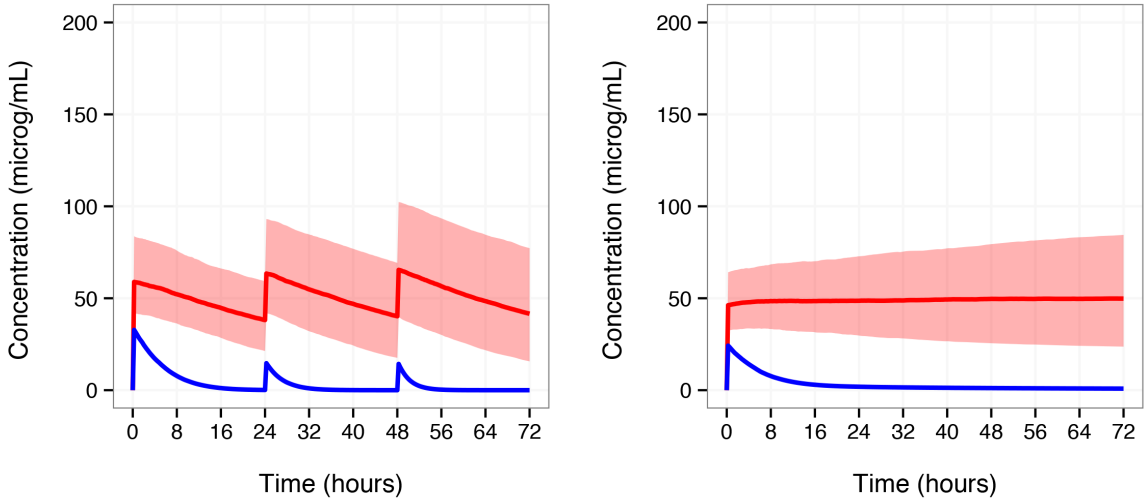
- How confident are you in your R programming skills?
- R simulation output can be evaluated with the simulation output provided by NONMEM® (or similar software) when given an equivalent population and dosing regimen (i.e., input dataset)
- If the model is from a published paper, compare results of your simulations with published figures

Tip 2: Perform Independent Checks

NONMEM® Simulations



R Simulations



- Simulation output should be approximately the same
- Precision increases with the number of simulations

Tip 2: Perform Independent Checks

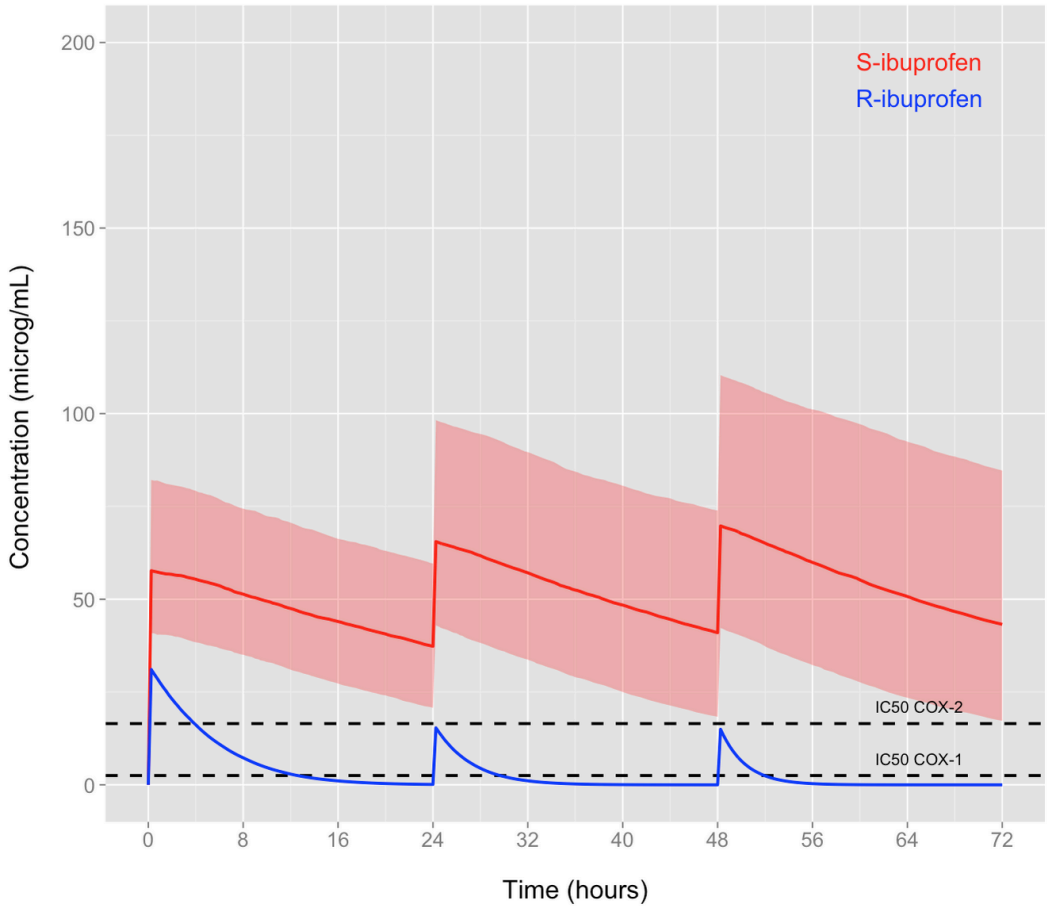
- Test a couple of different dosing regimens with the different software to be sure
  - Does the infusion switch off at the right time?
  - Achieve the same peak?
  - Achieve the same trough?
  - Same shape of the curve?
  - Same degree of variability (prediction intervals)?

Tip 3: Coding Differential Equations is "Easy"

- Using the deSolve and plyr/dplyr packages
- Three stage process:
    1. Create a population of individuals with their own individual parameter values for CL and V, for example
    2. Set up the model in differential equations
    3. Solve the set of differential equations at each specified time-point using each individual's own parameter and covariate values

Tip 4: Plot Median and Prediction Intervals

- An effective and simple way of graphically representing simulation results is by plotting the median (line) and prediction intervals (ribbon)



Tip 4: Plot Median and Prediction Intervals

- ggplot2
  - Use the `stat_summary` function to calculate and plot:
    - Median at each time-point as a line
    - Lower and upper percentiles at each time-point as a shaded ribbon
- ggplot2 + plyr
  - Calculate median and percentiles at each time-point prior to plotting using `ddply` from the plyr package
    - `geom_line` and `geom_ribbon` for plotting with ggplot2

Tip 5: Take it Slow when Converting to Shiny

- Build the user-interface first
    - Create the widgets for your model's input
- Build the server slowly
    - Build the first data frame and have it output to the user-interface – check the application is doing what you think
    - Start sequentially adding the other parts
    - You may find yourself running the Shiny application again and again to check your new additions

Tip 6: Control Application Speed

- Every time a widget changes, the linked `reactive` expression within `shinyServer` recognises that its stored values are outdated and they are re-evaluated to accommodate the new values
    - Even a small change in a slider causes the expression to be re-evaluated
- Solving a system of differential equations can be slow
    - Doing it repeatedly is even slower!

Part 1: Control Speed with Shiny functions

- Code outside `shinyServer` is run once on application initiation
    - Minimise redundant computation by separating `reactive` and non-reactive expressions
- `isolate` expressions linked with an `actionButton`
    - Control when computationally intense expressions are re-evaluated
        - i.e., data frames and linked output derived from solving differential equations

Part 1: Control Speed with Shiny functions

```
#Define user-input dependent functions for output
  shinyServer(function(input, output) {

#Reactive expression to generate a reactive data frame
#This is called whenever the input changes
  all.data <- reactive({

    input$UPDATE

    isolate({


     ####Code resulting in "sim.data"####



    })  #Brackets closing "isolate"expression

  sim.data <- as.data.frame(sim.data)

  })  #Brackets closing "reactive" expression

})  #Brackets closing "shinyServer" function
```

Input linked to an `actionButton` in ui.R called “UPDATE”

Shiny will not rebuild “sim.data” when input within `isolate` changes – only when input for “UPDATE” changes

Part 2: Increase Speed from R

- Compiling functions (using compiler)
- Parallel processing (using doParallel)
- Updated versions of packages or newer packages
- Coding differential equations and using differential equation solvers from other programming languages that can be adopted into R, i.e., C++
- Analytical solutions where possible

Tip 7: Design an Intuitive User-Interface

- Who is going to use or see your application?
- What is its intended use?
- It is easy to get inundated with widgets that crowd the user-interface
    - It is likely that not all widgets are required to be freely available to the user
- `conditionalPanel` allows for the user-interface to change appearance depending on user-input

Tip 7: Design an Intuitive User-Interface

```
#Selection box for dosing regimen
radioButtons("SELECT",
       "Dosing Regimen:",
       choices = list("Loading-Bolus-Bolus" = 1,
                      "Loading-Continuous Infusion" = 2),
       selected = 1),

  br(), #Add a blank break between widgets

  conditionalPanel(condition = "input.SELECT == 1",

#Slider input for dose
#Bolus dose for Loading-Bolus-Bolus regimen
sliderInput("BDOSE",
       "Bolus Dose at 24 and 48 hours (mg/kg):",
       min = 0,
       max = 40,
       value = 5,
       step = 1)

  ),  #Brackets closing "conditionalPanel"
```

Widget that controls the visibility of other widgets

Widget appears only when the "condition" is met

Software Versions used in Development of Example Application

Software/R Package Library	Version
R	3.1.2
RStudio	0.98.977
shiny	0.12.2
ggplot2	1.0.1
deSolve	1.12
plyr	1.8.3
compiler	3.1.2

Further Help on this Application Type

- Example script for coding population PK models in R available with the Shiny tutorial
  - Wojciechowski J, Hopkins AM, Upton RN. Interactive Pharmacometric Applications Using R and the Shiny Package. *CPT: pharmacometrics & systems pharmacology* 2015, 4(3): 1-14.