Introduction to the Shiny Framework for Pharmacometrics

Jessica Wojciechowski

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

20th May 2016

University of South Australia

Australian Centre for Pharmacometrics

Overview

- Demonstration of example Shiny applications
  - Example scripts for the second application are posted on GitHub
  - https://github.com/isop-phmx/studyGroup/issues/19
  - It is a pre-built Shiny application that we will make amendments to throughout this session
- What is Shiny?
- Why would a pharmacometrician want to use Shiny?
- Introduction to the Shiny framework
  - More advanced concepts can be addressed in later study group sessions

Distributions

Distribution type

Normal

Length of x:

0 1 10

0 1 2 3 4 5 6 7 8 9 10

Mean:

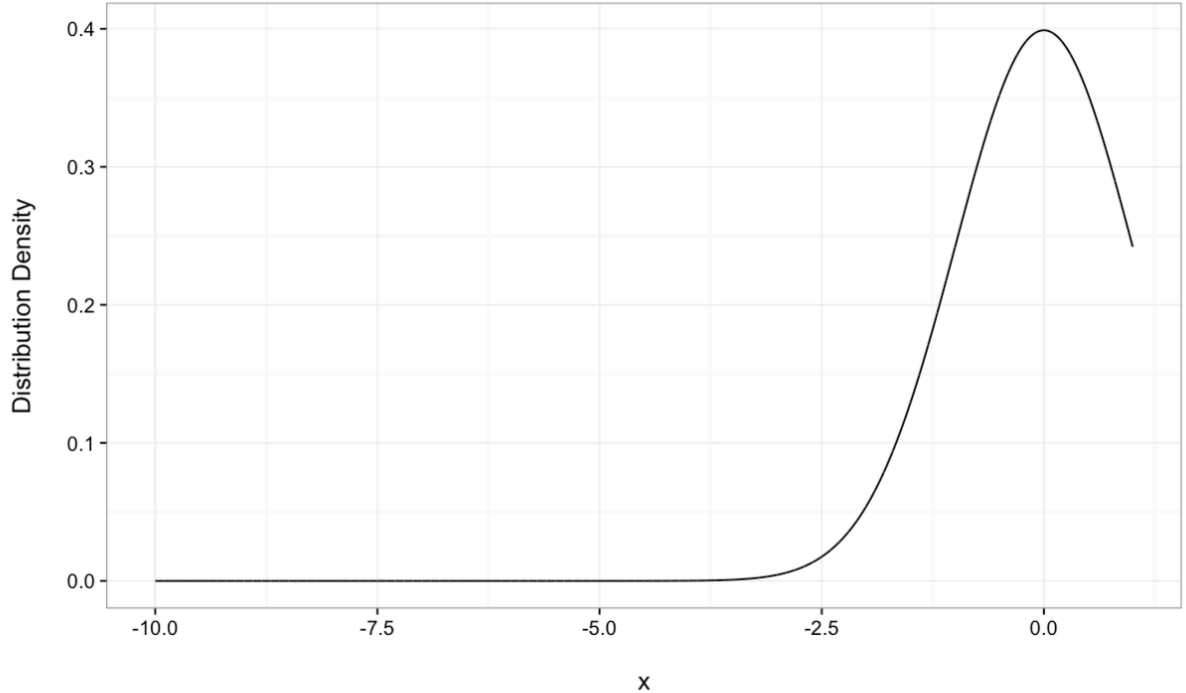
-10 0 10

-10 -8 -6 -4 -2 0 2 4 6 8 10

Standard Deviation:

0 1 10

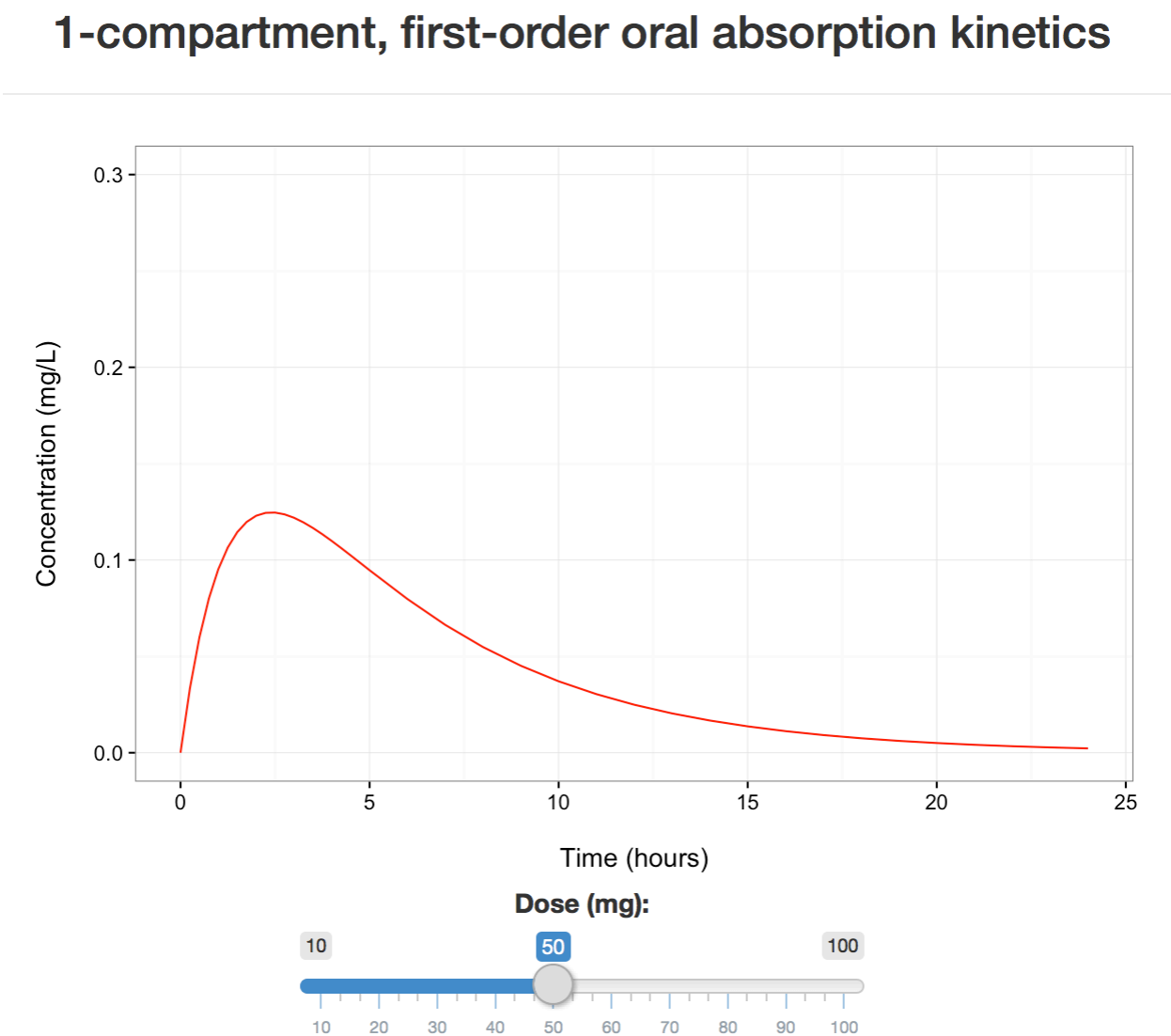
0 1 2 3 4 5 6 7 8 9 10



Shiny for R

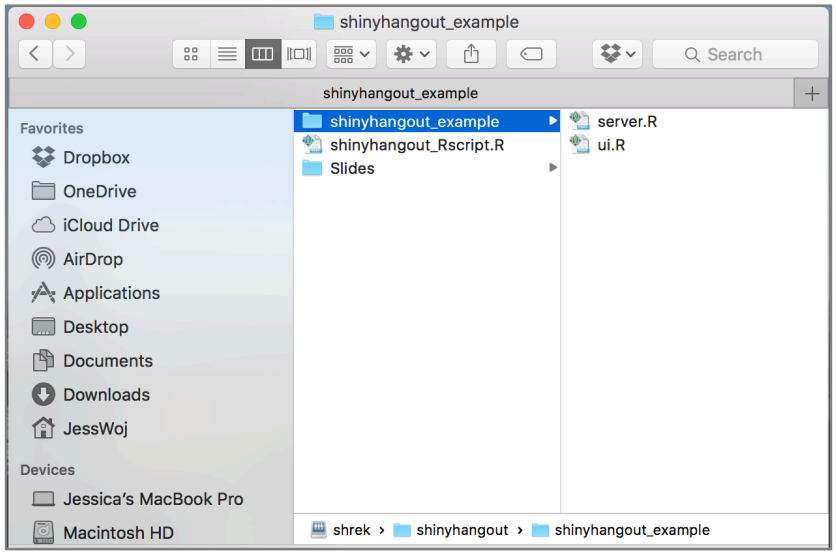
- Interactively show output for R programs
- Control in coding:
  - Appearance of application's user-interface
  - Generated output
- Share and view applications without an R installation or files containing R code
  - On another computer in a web-browser via the Internet

Example Application



Structure of a Shiny Application

- ui.R
  - Application's graphical user-interface
- server.R
  - Instructions for turning input into output to be displayed in the user-interface
- Need to be saved in the same folder



server.R | ui.R

Run App

- Run in Window
- Run in Viewer Pane
- ✓ Run External

```
#server.R script for shinyhangout_example application
#Calls reactive input from ui.R to simulate a concentration-time resulting profile
#-----------------------------------------------------------------------------
#Non-reactive objects/expressions, i.e., do not depend on reactiR
#Load package libraries
library(shiny)  #Package for creating user-interface for R program
library(ggplot2)  #Plotting

#Define sequence for time at which concentrations will be calculated
#Heavier sampling at the beginning of the sequence (every 15 minutes), then every hour after 5 hours since administration
TIME <- sort( unique(c(seq(from = 0,to = 5,by = 0.25),seq(from = 5,to = 24,by = 1))))

#-----------------------------------------------------------------------------
#Reactive objects/expressions, i.e., those that depend on values from the ui.R need to be within "shinyServer"
shinyServer(function(input, output) {  #Requires "input" and "output" objects
  output$concPlot <- renderPlot({  #"renderPlot" creates a "reactive" plot output object called "concPlot"

    #Assign the stored widget values from ui.R to objects in the "renderPlot" expression
    WT <- input$WT  #Call in the widget value for "WT", weight (kg)
    CRCL <- input$CRCL  #Call in the widget value for "CRCL", creatinine clearance (mL/min)
    if (input$DOSE == 1) {  #Call in the widget value for "DOSE"
      DOSE <- 50  #Where option "1" was the 50 mg dose
    } else {
      DOSE <- 100  #Where option "2" was the 100 mg dose
    }

    #Calculate pharmacokinetic parameters based on widget input and model's structure
    CL <- 15*(WT/70)^0.75*(CRCL/90)^-1.5  #Clearance, L/h, dependent on weight and creatinine clearance input
    V <- 20*(WT/70)  #Volume of distribution, L, dependent on weight input
    KA <- 0.2  #Absorption rate constant, h^-1
    #For each value of TIME (described earlier), simulate concentrations based on the above pharmacokinetic parameters
    #1-compartment, first-order oral absorption model
    CONC <- DOSE*KA/V*(KA-CL/V)*(exp(-CL/V*TIME)-exp(-KA*TIME))

    #Create the ggplot2 object for output
    plotobj <- ggplot()
```

1:1 (Top Level) R Script

Environment | History | Files

Console ~/

RStudio opens a new window

Application runs my bottom-right pane

Application runs in default web-browser

Error messages will appear here (console) when application is running

Install R packages

Plots | Packages | Help | Viewer

Install | Update | shiny

Name	Description	Version
shiny	Web Application Framework for R	0.13.2
shinydashboard	Create Dashboards with 'Shiny'	0.5.1
shinyjs	Perform Common JavaScript Operations in Shiny Apps using Plain R Code	0.5.2
shinystan	Interactive Visual and Numerical Diagnostics and Posterior Analysis for Bayesian Models	2.1.0
shinythemes	Themes for Shiny	1.0.1
miniUI	Shiny UI Widgets for Small Screens	0.1.1
rsconnect	Deployment Interface for R Markdown Documents and Shiny Applications	0.4.2.2

ui.R: Creating a user-interface

- Variety of customisable layouts and pre-built widgets
- Hierarchical structure
- Three levels for creating an interface:
  1. Layout function
     - `fluidPage, fixedPage`
  2. Positioning function
     - `fluidRow, fixedRow, sidebarLayout`
  3. Element
     - Widget functions
     - Reactive output functions
     - Headings, line breaks, images
- Functions of the same level are written as a string within their superior function, separated by “,”

ui.R: Shiny Widgets

- Interactive elements
- Allow users to explore different values or categories of parameters or variables
- Store values chosen by the user
  - That are involved in a cascade of functions in server.R to produce the displayed output
- Changing a widget, changes the resulting output object
- Examples:
  - Sliders
  - Selection boxes
  - Checkboxes
  - Radio buttons
  - Download button

ui.R: Displaying Reactive Output

- Output functions call R objects in server.R to the UI
- Examples:
  - `plotOutput`
  - `tableOutput`
  - `textOutput`
- Specific for the object in server.R
  - i.e. ggplot2 object called by `plotOutput`

server.R | ui.R | server.R | ui.R

Run App

```r
#ui.R script for shinyhangout_example appplication
#Defines widgets for reactive input for server.R and calls output objects from server.R and presents them in the user-interface
#-----------------------------------------------------------------------------------
#All user-interface elements need to be bound within a page formatting function, i.e., "fixedPage"
#This application places each element below the previous one in order from top to bottom
fixedPage(
  h3("1-compartment, first-order oral absorption kinetics"),  #Application title

  hr(),  #Horizontal separating line

  plotOutput("concPlot",width = 600),  #Concentration-time profile output

  sliderInput("DOSE",
              "Dose (mg):",
              min = 10,
              max = 100,
              step = 5,
              value = 50),  #Slider input for dose

  align = "center"  #Align all elements in the centre of the application

)  #Brackets closing "fixedPage"

```

server.R: Application Instructions

- Controls the processing of widget input to display output to the UI
- Can use your favourite R packages and their functions
- Structure of server.R code plays an important role:
  - Minimising redundant computation
  - Maximising application speed

server.R: Non-Reactive versus Reactive

- "Non-reactive" objects include:
  - Package libraries
  - Constant expressions, i.e., time
  - Loaded datasets
- Objects dependent on widget input are "reactive"
- Reactive objects need to be within the `shinyServer` function in server.R
  - They are called from ui.R to server.R by the widget's name:
    - `DOSE` in ui.R = `input$DOSE` in server.R
  - The value for the reactive object updates every time input from a widget changes
  - Code in `shinyServer` re-executes with every widget change
  - Code outside `shinyServer` is run once on application initiation

server.R: Sending Reactive Output to the UI

- Functions that create reactive objects to be sent to ui.R
  - `renderPlot`
  - `renderTable`
  - `renderText`
- Can contain code for the reactive object and code processing widget input
- Code within `shinyServer`
- Specific for the object type
  - If ggplot2 object is defined within `plotOutput` in ui.R
  - Then defined within `renderPlot` in server.R

ui.R | server.R

Run App

```r
#server.R script for shinyhangout_example application
#Calls reactive input from ui.R to simulate a concentration-time profile and plots the resulting profile
#-----------------------------------------------------------------------------------
#Non-reactive objects/expressions, i.e., do not depend on reactive widget input from ui.R
#Load package libraries
library(shiny)  #Package for creating user-interface for R program
library(ggplot2)  #Plotting

#Define custom ggplot theme
theme_bw2 <- theme_set(theme_bw(base_size = 14))

#Define sequence for time at which concentrations will be calculated
#Heavier sampling at the beginning of the sequence (every 15 minutes), then every hour after 5 hours since administration
TIME <- sort(unique(c(seq(from = 0,to = 5,by = 0.25),seq(from = 5,to = 24,by = 1))))

#-----------------------------------------------------------------------------------
#Reactive objects/expressions, i.e., those that depend on values from the ui.R need to be within "shinyServer"
shinyServer(function(input, output) {  #Requires "input" and "output" objects
  output$concPlot <- renderPlot({  #"renderPlot" creates a "reactive" plot output object called "concPlot"

    #Assign the stored widget values from ui.R to objects in the "renderPlot" expression
    DOSE <- input$DOSE  #Call in the widget value for "DOSE", dose (mg)
    #Other covariate information
    WT <- 70  #Weight (kg)
    CRCL <- 90  #Creatinine clearance (mL/min)

    #Calculate pharmacokinetic parameters based on widget input and model's structure
    CL <- 15*((WT/70)^0.75)*((CRCL/90)^1.5)  #Clearance, L/h
    V <- 20*(WT/70)  #Volume of distribution, L
    KA <- 0.2  #Absorption rate constant, h^-1
    #For each value of TIME (described earlier), simulate concentrations based on the above pharmacokinetic parameters
    #1-compartment, first-order oral absorption model
    CONC <- DOSE*KA/V*(KA-CL/V)*(exp(-CL/V*TIME)-exp(-KA*TIME))

    #Create the ggplot2 object for output
    plotobj <- ggplot()
    plotobj <- plotobj + geom_line(aes(x = TIME,y = CONC),colour = "red")
    plotobj <- plotobj + scale_x_continuous("\nTime (hours)")
    plotobj <- plotobj + scale_y_continuous("Concentration (mg/L)\n",lim = c(0,0.3))
    print(plotobj)  #This is the resulting object that will be sent to ui.R

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

"Non-reactive"

"Reactive"

Shiny for Pharmacometricians

- R is a programming and statistical language common to many pharmacometricians
- Flexible R language allows any of our models to be coded
- With the addition of Shiny:
  - Rapidly test dosing scenarios through simulation
  - Exploring the impact of a model's covariate effects
  - Identify initial parameter estimates prior to population modelling
  - Demonstrate properties of the model itself to a collaborating clinician
  - Teaching pharmacokinetic and pharmacodynamic principles to students
  - Share your work with others not familiar with or do not have an installation of R on their computer
  - Develop a prototype prior to formal web-app development

Useful Shiny Resources

- Shiny by RStudio
  - Shiny developers provide a tutorial series built with exercises, videos and discussion boards
  - http://shiny.rstudio.com/
- Stack Overflow
  - http://stackoverflow.com/questions/tagged/shiny
- GitHub
  - https://github.com/rstudio/shiny
- Shiny Google mailing list
  - https://groups.google.com/d/forum/shiny-discuss
- Commercially available book, *"Web Application Development with R using Shiny"* by Chris Beeley

More Shiny Resources

- Shiny Tutorial paper in CPT:PSP
  - Wojciechowski J, Hopkins AM, Upton RN. Interactive Pharmacometric Applications Using R and the Shiny Package. CPT: pharmacometrics & systems pharmacology. 2015;4(3):1-14.
- Shiny Tutorial from ACoP6
  - Building Pharmacometric Applications using R: R Shiny Tutorial. American Conference of Pharmacometrics (ACoP6) Annual Conference, October 2015: Arlington, VA, USA.
  - Available online: http://discuss.go-isop.org/t/building-pharmacometric-applications-using-r-an-online-r-shiny-tutorial/57

Concepts not Covered

- `reactive` and `isolate` expressions
  - http://shiny.rstudio.com/
- Storing application output
- Sharing Shiny applications
  - How to use RStudio's Shiny Server
    - http://www.shinyapps.io/
  - How to use Shiny Server and Shiny Server Pro
    - https://www.rstudio.com/products/shiny/shiny-server/
- Interactive documents and presentations with Shiny and RMarkdown
  - http://rmarkdown.rstudio.com/authoring_shiny.html
- Applications implementing population models
  - More R programming than Shiny – there are previous ISoP sessions covering these topics/packages

Concepts not Covered

- Fancier user-interface designs
  - How to incorporate other languages into a Shiny application
    - HTML, Css, Javascript
  - shinydashboard package for R: https://rstudio.github.io/shinydashboard/