An Interactive Tool to Visualize Results from Uncertainty Quantification

Matthew Isaac *

Abstract
The term 'uncertainty quantification' refers to a class of methods used to understand the ways that variations to inputs of a system can potentially change the end state of that system. This article outlines the implementation of an interactive tool to visualize the results of a 'two-dimensional' uncertainty quantification analysis. The interactive tool is a deployable web application built with the `R Shiny` framework. Users can create, adjust, and save custom visualizations to assist in interpreting and presenting results.

Key Words: Interactive Visualization; `shiny` R Package

1. Introduction

Uncertainty quantification is a methodological framework used in engineering analyses (Ewing et al., 2018). It is used to understand how variability within the parameters (i.e. inputs) to some system impact the end state of that system. Engineering analysts use uncertainty quantification to assess and find the balance between design sufficiency and design efficiency. This is particularly important in fields where large-scale prototypes, testing, and data collection are extremely expensive. Engineers in these types of applications are increasingly relying on computational simulations to assess system designs.

The following article outlines the implementation of a web application containing an interactive tool to visualize uncertainty quantification results. The visualization tool has been specifically designed to visualize results of a 'two-dimensional' (2D) uncertainty quantification analysis, as described by Ewing et al. (2018). The outline of this article will proceed as follows: In Section 2, a brief overview of the 2D uncertainty quantification algorithm is given. In Section 3, the software packages and methods used to create the visualization tool are described. In Section 4, features and capabilities of the resulting web application are described. In Section 5, challenges with the current implementation and future enhancements to the web application are discussed. The code written to generate the web application is included in the Appendix.

2. Uncertainty Quantification Overview

The following description of the uncertainty quantification algorithm is adapted from Ewing et al. (2018). First, a few terms and definitions will be described:

System Response Quantity (SRQ): A parameter of particular interest directly related to the engineering system in question. The SRQ is the output (i.e. prediction) from the engineering model, and it is this value for which we are performing the uncertainty quantification analysis.

Engineering Model: A mathematical model that defines the relationship between the parameters (model inputs) and SRQ (model output).

*Department of Mathematics and Statistics, Utah State University, Logan, UT 84322–3900, USA. E-mail: `matt.isaac@aggiemail.usu.edu`

Aleatory Uncertainty: Uncertainty resulting from randomness inherent to a given parameter. Gaining more knowledge about the parameter will not reduce the uncertainty of the parameter.

Epistemic Uncertainty: Uncertainty resulting from a lack of knowledge about a given parameter. Gaining more knowledge about the parameter could reduce the uncertainty of the parameter.

The first step of uncertainty quantification is to identify sources of uncertainty (usually associated with model parameters) and classify them as either aleatory or epistemic. This classification process has been somewhat debated in literature (Der Kiureghian and Ditlevsen, 2009), and will not be discussed here as it is outside the scope of this article. Once these uncertainties related to the model parameters have been identified and classified as aleatory or epistemic, the uncertainty for each parameter must be described. Traditionally, epistemic uncertainties have been described by an interval over which any value in the interval is equally likely, while aleatory uncertainties have been assigned probability distributions. Ewing et al. (2018) proposes that all uncertainties, aleatory or epistemic, be described using probability distributions. This debate will not be discussed here as it is, again, outside the scope of this article. Once these distributions and/or intervals have all been assigned, they are carried through the model using Monte Carlo techniques as described below.

Given a nested 'for loop' programming structure with two loops, let n denote the number of iterations of the outer loop, and let m denote the number of iterations of the inner loop. In the outer for loop, values for the epistemic uncertainty parameters are selected randomly from the intervals/distributions assigned. Upon entering the inner loop, values for the aleatory uncertainty parameters are randomly chosen from the distributions assigned. The values chosen for the parameters in both the outer loop and the inner loop are then used as inputs in the engineering model to calculate a value for the SRQ. This value is stored and the inner loop continues running for the rest of the $m - 1$ iterations. All of the m SRQ values calculated from the m iterations of the inner loop are used to calculate an empirical cumulative distribution function (CDF) of SRQ values and the CDF is stored. The outer loop then begins its second iteration, and new values for the empirical uncertainty parameters are chosen. The inner loop then runs, sampling values from the aleatory parameter distributions for another m iterations, producing another CDF. This process continues until the outer loop has run all of its n iterations.

At this point, there will be n empirical CDFs that have been calculated, representing various possible realized values of the SRQ. This collection of CDFs is often referred to as an 'ensemble'. Ewing et al. (2018) suggests constructing a probability box, or "P-box". This P-box is found by calculating a lower percentile (e.g. the 5th percentile) and an upper percentile (e.g. the 95th percentile) of the CDF ensemble. This P-box can then be interpreted in several ways, including (1) selecting a SRQ value and extracting a probability interval, and (2) selecting a probability value and extracting an SRQ interval.

3. Methods

The main objective in developing this interactive visualization tool was to assist analysts in visualizing and interpreting results from an uncertainty quantification analysis. Since the actual uncertainty quantification analysis can be carried out with greater speed and computational power elsewhere, this implementation does not include the Monte Carlo portion of the algorithm described in Section 2. Rather, the visualization tool is designed to receive and visualize data from such a simulation. The web application was built using

the `shiny R` package (Chang et al., 2018) and the code for this application is contained in two files. The first file, `ui.R` (short for 'user interface') contains the code that controls the layout of the various panels and panes in the application, as well as the placement of the user interface elements (i.e. text input boxes, sliders, etc.). The `server.R` file contains the computational code that performs calculations, data manipulation, and generates the plots. The source code is included in the Appendix.

3.1 Software Packages

The following software packages were used in the development of the web application:

R: The code for the web application was written in the `R` programming language (R Core Team, 2018).

RStudio: Rstudio (RStudio Team, 2016) is an integrated development environment that was used to both develop and deploy the web application.

shiny: The `shiny` package and framework (Chang et al., 2018) constitutes the backbone of this project. This package provides a way to create and deploy web applications through RStudio (RStudio Team, 2016). It also contains the implementations for all user-interface components (toggle buttons, check boxes, numeric inputs, sliders, etc.).

ggplot2: The plotting functionality of the `ggplot2` package (Wickham, 2016) was used to generate the actual visualization and to add, remove, or adjust components on the plot.

shinydashboard: The `shinydashboard` package (Chang and Borges Ribeiro, 2018) was used as an aesthetic wrapper to the web application to improve its professional appearance.

shinyalert: The `shinyalert` package (Attali and Edwards, 2018) was used to create a pop-up message when the user takes certain actions.

shinyWidgets: The `shinyWidgets` package (Perrier et al., 2019) was used to create custom toggle buttons.

Since the existing `R` packages already implemented most of the tools needed to generate the plot and implement user interface elements, the primary task on this project was to seamlessly combine elements from the software packages above to create a user-friendly interactive visualization tool.

4. Results

The web application is currently deployed at `https://misaac.shinyapps.io/UQViz/`. The following subsections outline the basic functionality of the application.

4.1 Data Source

The 'Data Source' tab contains the controls that allow the user to select either a sample data set (provided for convenience) or to upload their own data in a `.csv` file. If data from a file is to be used, the x-values over which the CDFs are evaluated should be the first column, and the CDF y-values in the following columns.

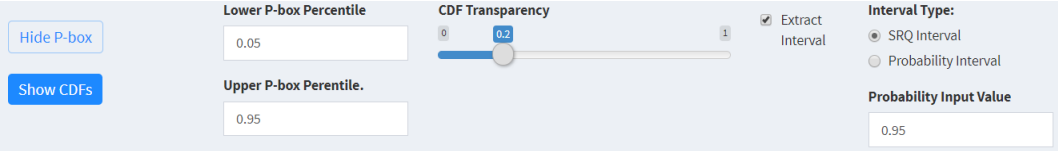

Figure 1: Screenshot from the web application of the visualization option control panel

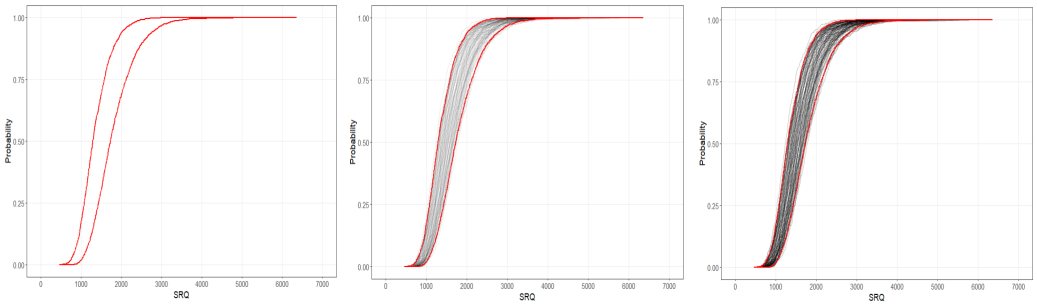

Figure 2: Visualization with P-box shown and CDFs hidden (left); Visualization with P-box and CDFs both shown with high transparency (center) and low transparency (right)

4.2 Visualization Options

Several visualization options are provided to allow customizable graphics to be created. A screenshot of the visualization option control panel is shown in Figure 1. The 'Show/Hide P-box' and the 'Show/Hide CDFs' buttons toggle back and forth between showing and hiding the red P-box and the black ensemble of CDFs. Note that showing the CDF ensemble may (depending on the size of the uploaded file) significantly increase plot rendering times.

The P-box lower and upper percentiles default to 0.05 and 0.95. Users can modify this by providing the desired lower and upper percentiles to be used in the visualization.

Moving the 'CDF Transparency' slider will make each individual CDF trace more or less transparent. A slider value of '1' corresponds to completely opaque, while a slider value of '0' corresponds to completely transparent. This mitigates the issue of overplotting that can often arise when hundreds or thousands of traces are shown on the same plot. Allowing for some transparency allows the user to visualize the density of regions within the CDF ensemble. Figure 2 shows plots with several possible visualization options selected.

To aid with interpretation, both probability intervals and SRQ intervals can be extracted from the P-box shown in the visualization, as described in Ewing et al. (2018). To extract a probability interval, the user selects an SRQ value at which a line is drawn vertically. The intersections of the vertical line with the P-box create the lower and upper endpoints for the probability interval. In a similar process, the user can also extract an SRQ interval by selecting a probability value at which a line is drawn horizontally. The intersections of the horizontal line with the P-box create the SRQ interval. This process is illustrated in Figure 3.

A title can be added to the plot and the x-axis label can be changed from the default label of 'SRQ'. This allows users to further customize the visualization to a specific domain or context.

Users can download and save a `.png` version of the current rendering of the visualization by providing a file name and clicking the 'Download' button.

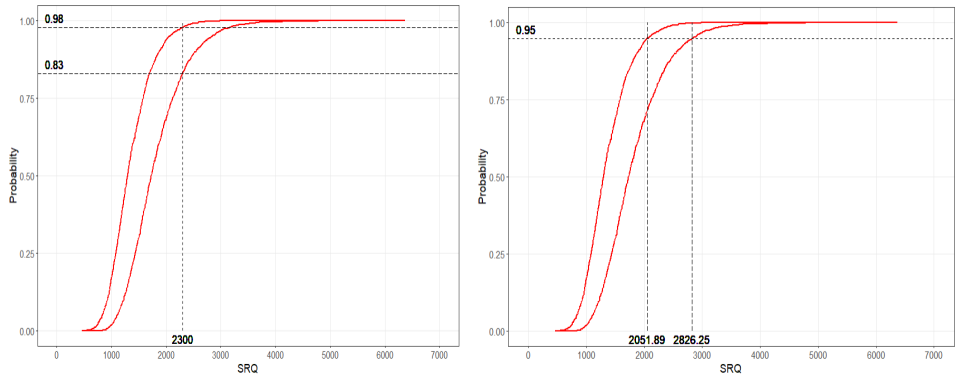

Figure 3: Extracting a probability interval (left); Extracting an SRQ interval (right)

5. Discussion

One main challenge with this visualization tool is the time that it takes to render the plot when the CDF ensemble is shown. The nature of uncertainty quantification and this kind of visualization leads to there being hundreds or even thousands of traces on a single plot. If the CDF ensemble is hidden, plot rendering times are almost immediate. On the other hand, if the CDF ensemble is shown on the plot, the user may have to wait anywhere from 5 to 35 seconds (depending on the size of the data set being used) for the plot to re-render. In addition, due to the way the interactive features are implemented through the `shiny` package, the plot completely re-renders when user inputs change. Thus, when a visualization option is adjusted and the CDFs are currently shown, the user may experience a lengthened rendering time. This leads to a less 'interactive' feel for the user. No immediate or direct solution has been found for this issue yet. However, an effective workaround has been suggested: If the user wishes to display the CDF ensemble on the final version of the plot, it may save time to adjust all visualization options and controls while the CDFs are hidden, keeping the plot rendering times to a reasonable length. Then, as the last step, show the CDF ensemble. This way, the user will only have to wait once for the lengthened rendering time.

Future work in developing this visualization tool may involve researching other methods for visualizing uncertainty, such as the methods described in Jackson (2008), and implementing several methods for visualization. This would allow users to choose from various visualization methods within the web application to create a graphic that is best suited to a given context. Another extension of this visualization tool could be to use it in a self contained UQ tool, where users could perform the entire UQ algorithm from start to finish. For example, users could define the engineering model to be used, define parameters as aleatory or epistemic, carry out the UQ algorithm, and visualize the results. However, as the web application now stands, it can be used to assist engineers and analysts in visualizing, understanding, and interpreting results from uncertainty quantification analyses.

References

Attali, D., Edwards, T., 2018. shinyalert: Easily Create Pretty Popup Messages (Modals) in 'Shiny'. R package version 1.0 (https://CRAN.R-project.org/package=shinyalert).

Chang, W., Borges Ribeiro, B., 2018. shinydashboard: Create Dashboards with 'Shiny'. R package version 0.7.1 (https://CRAN.R-project.org/package=shinydashboard).

Chang, W., Cheng, J., Allaire, J., Xie, Y., McPherson, J., 2018. shiny: Web Application Framework for R. R package version 1.2.0 (https://CRAN.R-project.org/package=shiny).

Der Kiureghian, A. D., Ditlevsen, O., 2009. Aleatory or Epistemic? Does it Matter? Structural Safety 31 (2), 105–112.

Ewing, M., Liechty, B. C., Black, D., 2018. A General Methodology for Uncertainty Quantification in Engineering Analyses Using a Credible Probability Box. Journal of Verification, Validation, and Uncertainty Quantification 3 (2), https://doi.org/10.1115/1.4041490.

Jackson, C. H., 2008. Displaying Uncertainty With Shading. The American Statistician 62 (4), 340–347.

Perrier, V., Meyer, F., Granjon, D., 2019. shinyWidgets: Custom Inputs Widgets for Shiny. R package version 0.4.8 (https://CRAN.R-project.org/package=shinyWidgets).

R Core Team, 2018. R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, Vienna, Austria, (https://www.R-project.org/).

RStudio Team, 2016. RStudio: Integrated Development Environment for R. RStudio, Inc., Boston, MA, (http://www.rstudio.com/).

Wickham, H., 2016. ggplot2: Elegant Graphics for Data Analysis. Springer-Verlag New York, (http://ggplot2.org).

A. Appendix

A.1 Source Code: `ui.R`

```
library(shiny)
library(shinydashboard)
library(plotly)
library(shinyalert)
library(shinyWidgets)

cranURL <- "https://cran.r-project.org/web/packages/"

dashboardPage(
  dashboardHeader(
    title = "UQViz"
  ),
  dashboardSidebar(
  ),
  dashboardBody(
    useShinyalert(),
    tabsetPanel(
      tabPanel("Data Source",
               fluidRow(
                 column(
                   width = 3,
                   radioButtons(inputId = "radioDataSource",
                                label = "Data Source",
                                choices = c("Sample Data",
                                            "File Upload"),
                                selected = "Sample Data"),
                   uiOutput(outputId = "fileInput")
                 )
               )

      ),
      tabPanel("Visualization Options",
               fluidRow(
                 column(
                   width = 2,
                   offset = 0,
                   verticalLayout(
                     br(),
                     uiOutput(outputId = "pbxBttn"),
                     br(),
                     uiOutput(outputId = "cdfBttn")

                   )

                 ),
                 column(
                   width = 2,
                   offset = 0,
                   numericInput(inputId = "pboxLower",
                                label = "Lower P-box Percentile",
                                value = 0.05,
                                width = 250),
                   numericInput(inputId = "pboxUpper",
                                label = "Upper P-box Perentile.",
                                value = 0.95,
                                width = 250)
                 ),
                 column(
                   width = 3,
                   offset = 0,
                   sliderInput(inputId = "sliderAlpha",
                               label = "CDF Transparency",
                               value = 0.2,
                               min = 0,
                               max = 1,
                               step = 0.1,
```

```
                            ticks = FALSE)
            ),
            column(
              width = 1,
              offset = 0,
              checkboxInput(inputId = "extractInt",
                            label = "Extract Interval",
                            value = FALSE)

            ),
            column(
              width = 2,
              uiOutput(outputId = "extractTypeInput"),
              uiOutput(outputId = "valInput")
            )
          )

  ),
  tabPanel("Labeling",
           fluidRow(
             column(
               width = 2,
               textInput(inputId = "titleInput",
                         label = "Title",
                         placeholder = "Plot Title",
                         value = "")
             ),
             column(
               width = 2,
               textInput(inputId = "xaxisInput",
                         label = "X axis Label",
                         placeholder = "X-axis",
                         value = "SRQ") #,
               # textInput(inputId = "yaxisInput",
               #           label = "Y axis Label",
               #           placeholder = "Y-axis",
               #           value = "Probability")
             )
           )
  ),
  tabPanel("User Guide",
           h3("Overview"),
           h5("UQViz is an interactive tool that can be used
              to create meaningful visualizations of
              '2D Uncertainty Quantification'
              (for more information on 2D UQ, see ",
              tags$a(
                href = paste0("http://verification.asmedigitalcollection",
                              ".asme.org",
                              "/article.aspx?articleid=",
                              "2703289&resultClick=1"),
                     "this article",
                     target = "_blank"),
              "). Users can select data source and adjust various
              visual components and labels. The title and axis
              label can also be customized, and the visualization
              can be downloaded and saved."),
           h3("Data Source"),
           tags$p("The 'Data Source' tab contains controls for
                  the user to select the source from
                  which data for the visualization will be obtained.
                  If ",
                  tags$b('Sample Data'),
                  "is selected, a small built-in data set is used as
                  the data source for the
                  visualization. This feature is provided for convenience,
                  and can be used for experimentation and
                  exploration. If ",
                  tags$b('File Upload'),
                  "is selected, users can upload a ",
                  tags$code(".csv"),
```

```
                "file containing
                data to be used in the visualization. This ",
                tags$code(".csv"),
                "file should be formatted such that the 'x' values over
                which the CDFs are evaluated
                are in the first column, and the CDFs are
                in the following columns."
        ),
        h3("Visualization Options"),
        tags$p("Several visualization options are provided to
                allow customizable graphics to be
                created. The ",
                tags$b("Show/Hide P-box"),
                " and the ",
                tags$b("Show/Hide CDFs"),
                " buttons toggle
                back and forth between showing and
                hiding the red P-box and the black ensemble of CDFs.
                Note that showing the CDF
                ensemble may (depending on the size of the uploaded
                file) significantly increase
                plot rendering times. If the user wishes to
                display the CDFs on the final version
                of the plot, it is suggested to adjust all
                visualization options and controls
                while the CDFs are hidden, keeping the plot
                rendering times to a reasonable
                length. Then, as the last step, show the CDF
                ensemble. This way, the user will only
                have to wait once for the lengthened rendering time."),
        tags$p("The ",
                tags$b("Lower P-box Percentile"),
                " and ",
                tags$b("Upper P-box Percentile"),
                "values can also be adjusted. This will
                shift the boundaries of the P-box on
                the visualization."),
        tags$p("Adjusting the ",
                tags$b("CDF Transparency"),
                " slider will adjust the
                transparency of each CDF in the CDF ensemble.
                A value of 0 corresponds to
                completely transparent, while a value
                of 1 corresponds to completely opaque.
                This control will only affect the visualization
                when the CDFs are being shown.
                This feature can be useful when there is a
                high degree of overplotting due to
                a large number of CDFs. Allowing transparency
                on the CDFs can assist in
                visualizing high-density regions on the plot."),
        tags$p("When the ",
                tags$b("Extract Interval"),
                " box is checked,
                several more controls appear.
                Extracting an interval is one way that
                P-boxes can be interpreted in practice.
                Users can decide whether to extract an ",
                tags$b("SRQ Interval"),
                " or a ",
                tags$b("Probability Interval"),
                ". If extracting an SRQ interval, users
                will need to select a probablity value; if
                extracting a probability interval, users
                will need to select an SRQ value.
                Labels and dotted lines will be displayed
                on the visualization, showing the
                probability or SRQ value selected and the
                corresponding interval."),
        h3("Labeling"),
        tags$p("Users also have the capability to choose
```

```
                    a custom ",
                    tags$b("Title"),
                    " and a custom ",
                    tags$b("X axis Label"),
                    "to futher customize the plot to a
                    specific context or application. Since this
                    visualization was designed to always show
                    probability on the y axis, the y axis label
                    cannot be changed."),
            h3("Saving a Plot"),
            tags$p("Users can enter a ",
                    tags$b("File Name"),
                    " and can click the " ,
                    tags$b("Download"),
                    " button to download and save a ",
                    tags$code(".png"),
                    " file of the current version of the visualization.
                    These controls are located below the plot.
                    Note that the chosen
                    file name should omit the ",
                    tags$code(".png"),
                    " file extension.")

    ),
    tabPanel("References",
             tags$h4("The following software packages
                      were used in development: "),
             tags$div(tags$ul(
               tags$li(tags$span(tags$a(
                 href = "https://www.R-project.org/",
                 target = "_blank",
                 "R"))),
               tags$li(tags$span(tags$a(
                 href = "http://www.rstudio.com/",
                 target = "_blank",
                 "RStudio"))),
               tags$li(tags$span(tags$a(
                 href = paste0(cranURL, "shiny/index.html"),
                 target = "_blank",
                 "shiny"))),
               tags$li(tags$span(tags$a(
                 href = paste0(cranURL, "ggplot2/index.html"),
                 target = "_blank",
                 "ggplot2"))),
               tags$li(tags$span(tags$a(
                 href = paste0(cranURL, "shinyWidgets/index.html"),
                 target = "_blank",
                 "shinyWidgets"))),
               tags$li(tags$span(tags$a(
                 href = paste0(cranURL, "shinydashboard/index.html"),
                 target = "_blank",
                 "shinydashboard"))))
             )
    )
  ),
  fluidRow(
    box(
      width = 12,
      height = 625,
      plotOutput("uqPlot",
                 height = 475),
      fluidRow(
        column(
          width = 3,
          textInput(inputId = "pngName",
                    value = "",
                    placeholder = "file name",
                    label = "File Name (omit file extension)",
                    width = 200),
          downloadButton(outputId = 'savePng')
        )
```

```
        )
      )
    )
  )
)
```

A.2 Source Code: `server.R`

```
# Load needed packages
library(shiny)
library(ggplot2)
library(plotly)
library(shinyalert)
library(shinyWidgets)

## Increase max file upload size
options(shiny.maxRequestSize = 15 * 1024 ^ 2)

################################################
# Functions ####################################
################################################

# Function to calculate p-box lower and upper bounds
calc_pbox <- function(data, p_upper, p_lower){
  pbox <- apply(values$data,
                1,
                FUN = quantile,
                probs = c(p_lower, p_upper))

  pbox <- t(pbox)

  values$pbox <- data.frame(x = values$data[, 1],
                            lower = pbox[, 1],
                            upper = pbox[, 2])
  return(pbox)
}

# function to calculate the SRQ value for a given probability level
calc_srq <- function(prob_in){
  srq_l <- values$pbox[which.min(abs(values$pbox$lower - prob_in)), ]$x
  srq_u <- values$pbox[which.min(abs(values$pbox$upper - prob_in)), ]$x

  return(c(srq_l, srq_u))
}

# function to calculate the probability value for a probability level
calc_probs <- function(srq_in){
  prob_l <- values$pbox[which.min(abs(values$pbox$x - srq_in)), ]$lower
  prob_u <- values$pbox[which.min(abs(values$pbox$x - srq_in)), ]$upper

  return(c(prob_l, prob_u))
}

################################################
# Reactive Values###############################
################################################

values <- reactiveValues()

################################################
# Read in Sample Data ##########################
################################################
cdf_arr <-  read.csv("data/cdfs.csv", header = FALSE)
cdf_arr <-  t(cdf_arr)
cdf_df <-  data.frame(cdf_arr)

values$valmax <- max(cdf_df[, 1])
values$valmin <- min(cdf_df[, 1])
```

```
values$userin <- NULL
values$showcdf <- FALSE
values$showpbx <- TRUE

##############################################
# Server Logic ###############################
##############################################
shinyServer(function(input, output) {

  # toggle button to show/hide cdf ensemble
  observeEvent(input$cdfToggle, {
    if (values$showcdf == FALSE){
      shinyalert(
        title = "",
        callbackR = function(x){
          values$showcdf <- x
        },
        text = paste("Showing the CDF ensemble on the plot may significantly",
                     "increase plot render times. Do you want to continue?"),
        showCancelButton = TRUE,
        confirmButtonText = "Show CDFs",
        cancelButtonText = "Don't Show CDFs",
        confirmButtonCol = "#52D755",
        type = "info"
      )
    } else {
    values$showcdf <- FALSE
    }
  })

  observeEvent(input$pbxToggle, {
    if (values$showpbx == FALSE){
      values$showpbx <- TRUE
    } else {
      values$showpbx <- FALSE
    }
  })

  # toggle button to show/hide p-box
  output$pbxBttn <- renderUI({
    if (values$showpbx == TRUE){
      lbl <- "Hide P-box"
      sty <- "bordered"
    } else {
      lbl <- "Show P-box"
      sty <- "simple"
    }

    actionBttn(inputId = "pbxToggle",
               label = lbl,
               style = sty,
               size = "sm",
               color = "primary")
  })

  output$cdfBttn <- renderUI({
    if (values$showcdf == TRUE){
      lbl <- "Hide CDFs"
      sty <- "bordered"
    } else {
      lbl <- "Show CDFs"
      sty <- "simple"
    }

    actionBttn(inputId = "cdfToggle",
               label = lbl,
               style = sty,
               size = "sm",
               color = "primary")
  })
```

```
output$uqPlot <- renderPlot({
  # Use selected data source
  if (input$radioDataSource == "Sample Data"){
    values$data <- cdf_df
  } else {
    req(input$fileIn)
    df <- read.csv(input$fileIn$datapath,
                   header = FALSE)
    df <- data.frame(df)
    values$data <- df
  }

  # set up initial plot
  req(values$data)
  plt <- ggplot(data = values$data,
                aes(values$data[, 1]))

  # plot cdf ensemble if user selects
  if (values$showcdf == TRUE){
    for (i in names(values$data)[-1]) {
      plt <- plt +
        geom_line(aes_string(y = i,
                             color = shQuote("CDFs")),
                  alpha = input$sliderAlpha)
    }
    plt <- plt
  }

  # get nice breaks for x-axis
  px <- pretty(values$data[, 1])

  # Labels and additional plot formatting
  plt <- plt +
    xlab(input$xaxisInput) +
    # ylab(input$yaxisInput) +
    ylab("Probability") +
    scale_x_continuous(breaks = px,
                       limits = range(px)) +
    ggtitle(input$titleInput) +
    theme_bw() +
    theme(plot.title = element_text(hjust = 0.5,
                                    size = 18,
                                    face = "bold"),
          panel.grid.minor = element_blank(),
          axis.text = element_text(size = 12),
          axis.title = element_text(size = 14))

  # show p-box if user selects
  if (values$showpbx == TRUE){
    pbox <- calc_pbox(values$data,
                      (1 - input$pboxLower),
                      (1 - input$pboxUpper))
    plt <- plt +
      geom_line(aes(y = pbox[, 1], col = "Pbox"), lwd = 1) +
      geom_line(aes(y = pbox[, 2], col = "Pbox"),  lwd = 1) +
      scale_colour_manual(name = "", values = c("Pbox" = "red"))
  }

  # Calculate and show extracted interval on plot
  # if user selects
  if (input$extractInt == TRUE){
    req(input$valIn)

    # if interval is an SRQ interval
    if (input$extractType == "SRQ Interval"){
      if (input$valIn < 0){
        values$userin <- 0
      } else if (input$valIn > 1) {
        values$userin <- 1
      } else {
        values$userin <- input$valIn
```

```
      }
      srq_vec <- calc_srq(values$userin)
      plt <- plt +
        geom_hline(yintercept = values$userin, lty = 2) +
        geom_segment(mapping = aes(x = srq_vec[1],
                                   y = 0,
                                   xend = srq_vec[1],
                                   yend = 1),
                     lty = "longdash") +
        geom_segment(mapping = aes(x = srq_vec[2],
                                   y = 0,
                                   xend = srq_vec[2],
                                   yend = 1),
                     lty = "longdash") +
        geom_text(aes(srq_vec[1], 0.05,
                      label = round(srq_vec[1], 2),
                      vjust = 3,
                      hjust = 0.5),
                  size = 5) +
        geom_text(aes(srq_vec[2], 0.05,
                      label = round(srq_vec[2], 2),
                      vjust = 3,
                      hjust = 0.5),
                  size = 5) +
        geom_text(aes(min(values$data[, 1]),
                      values$userin,
                      label = round(values$userin, 2),
                      vjust = -0.5,
                      hjust = 2),
                  size = 5)
    } else {
      # if interval is a probability interval
      req(input$valIn)
      if (input$valIn < values$valmin){
        values$userin <- values$valmin
      } else if (input$valIn > values$valmax) {
        values$userin <- values$valmax
      } else {
        values$userin <- input$valIn
      }
      prob_vec <- calc_probs(values$userin)
      plt <- plt +
        geom_segment(mapping = aes(x = values$userin,
                                   y = 0,
                                   xend = values$userin,
                                   yend = 1),
                     lty = 2) +
        geom_hline(yintercept = prob_vec[1],
                   lty = "longdash") +
        geom_hline(yintercept = prob_vec[2],
                   lty = "longdash") +
        geom_text(aes(min(values$data[, 1]),
                      prob_vec[1],
                      label = round(prob_vec[1], 2),
                      vjust = -0.5,
                      hjust = 2),
                  size = 5) +
        geom_text(aes(min(values$data[, 1]),
                      prob_vec[2],
                      label = round(prob_vec[2], 2),
                      vjust = -0.5,
                      hjust = 2),
                  size = 5) +
        geom_text(aes(values$userin,
                      0.05,
                      label = round(values$userin, 2),
                      vjust = 3,
                      hjust = 0.5),
                  size = 5)
    }
  }
```

```
  if (values$showpbx && values$showcdf){
    plt <- plt +
      scale_color_manual(name = "",
                         values = c("CDFs" = "black",
                                    "Pbox" = "red")) +
      guides(colour =
               guide_legend(override.aes =
                              list(linetype = c(1, 1),
                                   lwd = c(0.5, 1)))) +
      theme(legend.text = element_text(size = 12))
  } else if (values$showpbx){
    plt <- plt +
      scale_color_manual(name = "",
                         values = c("Pbox" = "red")) +
      theme(legend.text = element_text(size = 12))
  } else if (values$showcdf){
    plt <- plt +
      scale_color_manual(name = "",
                         values = c("CDFs" = "black")) +
      theme(legend.text = element_text(size = 12))
  }

  values$plot <- plt

  plt
 })

output$extractTypeInput <- renderUI({
  if (input$extractInt == TRUE){
    radioButtons(inputId = "extractType",
                 label = "Interval Type: ",
                 choices = c("SRQ Interval",
                             "Probability Interval"))
  }
})


output$valInput <- renderUI({
  if (input$extractInt == TRUE){
    if (input$extractType == "Probability Interval"){
      lbl <-  "SRQ Input Value"
      val <-  round(quantile(values$data[, 1], 0.1), 0)
    } else {
      lbl <- "Probability Input Value"
      val <- 0.95
    }
    numericInput(inputId = "valIn",
                 label = lbl,
                 value = val)
  }
})

output$fileInput <- renderUI({
  if (input$radioDataSource == "File Upload"){
    fileInput(inputId = "fileIn",
              label = "Upload .csv File")
  }
})

# help from https://stackoverflow.com/questions
# /14810409/save-plots-made-in-a-shiny-app
output$savePng <- downloadHandler(
  filename = function(){
    paste(input$pngName,
          ".png",
          sep = "")
  },
  content = function(file){
    ggsave(file,
           plot = values$plot,
```

```
            device = "png",
            width = 12)
    }
  )
})
```