**Dynamic UI’s (repeating lists of UI elements)**

General Function

This function here is the UI code for the repeatable Module. Notice ns <- NS(id) which makes a new namespace for this module so that you can access the id.

listOutputUI <- function(id, type){

ns <- NS(id)

if (type == "Int"){

tagList(

h3(paste0("Input: ", id, sep = "")),

numericInput(ns("numericText"),"Numeric Input", 10, width = '100%'),

hr()

)}}

UI

uiOutput("dynamicUIInputList")

Server

output$dynamicUIInputList <- renderUI({

lapply(1:values$numRows, function(i) {

listOutputUI(values$names[i], values$types[i] )}

})

The callModule function supplies the first three arguments of the Shiny Module’s function inputs - the input, output, and session. Additional arguments supplied by the user in the callModule function are passed to the specific shiny module that is called. There can be additional helper functions that are a part of a shiny module.

observe({

lapply(1:values$numRows, function(i) {

callModule(listOutput, values$names[i], i)

})

})

ModID here, when added to values$userInput, makes a named list. ModID represents the UI’s “number” (because if you have more than one, they are in a list of number)

listOutput <- function(input, output, session, modID){

observeEvent(input$numericText,{

values$userInput[modID] <- input$numericText

})}

**Dynamic UI’s (repeating lists of UI elements)**

General Function

This function here is the UI code for the repeatable Module. Notice ns <- NS(id) which makes a new namespace for this module so that you can access the id.

editInputUI <- function(name, type){

ns <- NS(name)

tagList(

textInput(name, label = NULL, placeholder = name),

#selectizeInput(name, label = NULL, options = list(placeholder = type)),

hr()

)

}

UI

uiOutput("editInputDynamicModule")

Server

Output$editInputDynamicModule <- renderUI({

lapply(1:values$numRows, function(i) {

editInputUI(values$names[i], values$types[i] )}

})

The callModule function supplies the first three arguments of the Shiny Module’s function inputs - the input, output, and session. Additional arguments supplied by the user in the callModule function are passed to the specific shiny module that is called. There can be additional helper functions that are a part of a shiny module.

observe({

lapply(1:values$numRows, function(i) {

callModule(listOutput, values$names[i], i)

})

})

listOutput <- function(input, output, session, modID){

observeEvent(input$numericText,{

values$userInput[modID] <- input$numericText

})}