

Make-Changes

Modify mizerShiny

Here contains information about how to load in your own `mizerParam` objects and how to add your own plotting functions.

Load new mizerParams and mizerSims

Firstly, the `mizerShiny()` function itself is coded for `mizer` models to be quickly loaded. It contains the argument `params_file` that allows you to load any `mizerParams` file, this is defaulted as `params.rds`. The `mizerParam` object is loaded from the folder 'inst/Including/mizerParam'.

```
mizerShiny(params_file = "myparams.rds")
```

As the app relies on comparison with 'base' `mizerSim` files, you may want to change these. The `sp_sim_file` argument provides the 'base' `mizerSim` object for the 'Single Species' section, and the `fish_sim_file` argument provides the same for the 'Fishery Strategy' section. These are loaded from the folder 'inst/Including/mizerSim'.

```
mizerShiny(  
  params_file = "yourparams.rds",  
  sp_sim_file = "yourspSim.rdata",  
  fish_sim_file = "yourfishSim.rdata" # optional  
)
```

The `REPLACE` argument will save the `mizerParams` and `mizerSim` files that you have provided to the respective folders. Therefore, it can be loaded with only `mizerShiny()`, making it quicker to use.

```
mizerShiny(params_file = "yourparams.rds", REPLACE = TRUE)
```

```
#Then next time you call it, or if you provide the file/package to others  
#This will run yourparams.rds instead of the default params.rds
```

```
mizerShiny()
```

```
#The same will work if you provide mizerSim files.
```

```
mizerShiny(  
  params_file = "yourparams.rds",  
  sp_sim_file = "yourspSim.rdata",  
  fish_sim_file = "yourfishSim.rdata",  
  REPLACE = TRUE)
```

```
#Then calling mizerShiny() will load your mizerParams and mizerSim files
```

```
#To get back to the params.rds default sims, either reinstall the package or repeat the process with th
```

The process outlined here is also discussed in the `mizerShiny()` documentation.

The amount of time that a User can project forward, set by the 'Time Range' slider in the app, is changed

depending on the `mizerSim` object provided. It is set to whatever the time is in the `mizerSim`, minus 2, then divided by two. It is required for the base simulation to be longer than the User set sim, as the ‘Species’ and ‘Guild’ tabs include bars which are 2x the Users chosen ‘Time Range’.

Add new plotting functions.

Firstly, you must have your own function, that accepts two `mizerSim` objects and two time arguments. One `mizerSim` object is user ran, the second is the ‘base’ which we compare against. The values plotted are within this time range. The function must output a ggplot object.

To put your code in, change the `server` code in *mizerShinyApp.R*. Find the section that you wish to change, either following `bioSimData`, `mortSimData` or `fishSimData`, corresponding to the ‘Single Species’ Biomass and Mortality sections, and the ‘Fishery Strategy’ section. There will be a repeated section of `output` code where each repeated chunk looks something like this.

```
output$fishspeciesPlot <- renderPlotly({
  req(fishSimData())

  win <- fish_win1()

  ggplotly(
    plotSpeciesWithTimeRange(
      fishSimData()$sim1,
      fishSimData()$unharv,
      win$start, win$end
    )
  )
})
```

Add you function in place of `plotSpeciesWithTimeRange` and change the output ID, here `fishspeciesPlot`. The `fish_win1()` code reactively takes the user input of the ‘Time Range’ in ‘Fishery Strategy’ tab and will rerun this section of code, so keep `win <- fish_win1()` to preserve this reactivity. The `win$start` and `win$end` are the start and end values for the time that is plotted. In the ‘Single Species’ section, the time inputs are direct from the `input$` provided by the ‘Time Range’ slider, so copy this functionality from a similar code chunk.

Now you need to add a section in the app for the plot to load to.

Navigate to the UI section (faster way is to find an output ID in the same output sequence that you have just added to, and CTRL+F). There will be a section that looks like below, for the biomass, mortality and fishery strategy sections. Wherever you are adding, find the equivalent.

```
div(
  class = "plot-card",
  style = "flex: 4.5; height:50vh; display:flex; flex-direction:column; overf",
  tabsetPanel(
    id = "fishy_plots",
    tabPanel(
      title = "Species",
      div(style = "flex:1; display:flex;",
        plotlyOutput("fishspeciesPlot", height = "100%", width = "100%")
      )
    ),
    tabPanel(
      title = "Yield",
      div(style = "flex:1; display:flex;",
        plotlyOutput("yieldPlot", height = "100%", width = "100%")
      )
    )
  )
)
```

```

    )
  ),
  tabPanel(
    title = "Size",
    div(style = "flex:1; display:flex;",
        plotlyOutput("fishsizePlot", height = "100%", width = "100%")
    )
  ),
  if (app_exists("Including", "guilds_information", "checkGuilds", "guildpa
    tabPanel(
      title = "Guild",
      div(style = "flex:1; display:flex;",
          plotlyOutput("fishguildPlot", height = "100%", width = "100%")
      )
    )
  },
  tabPanel(
    title = "Spectra",
    div(style = "flex:1; display:flex;",
        plotlyOutput("spectrumPlot", height = "100%", width = "100%")
    )
  ),
  tabPanel(
    title = "Diet",
    div(style = "height:50vh; display:flex;",
        plotlyOutput("fishdietsingleplot", height = "100%", width = "100%")
    )
  ),
  if (app_exists("Including", "Nutrition", "checkNutrition", "nutrition.csv
    tabPanel(
      title = "Nutrition",
      div(style = "flex:1; display:flex;",
          plotlyOutput("nutritionplot", height = "100%", width = "100%")
      )
    )
  }
)
)

```

Now copy a `tabPanel` argument, and replace the ID with your new ID.