

/demo

Large data with arrow



/scalability

The challenge with simple Shiny

ui.R

```
# UI.R
fluidPage(
  h4(title),
  shiny::sidebarLayout(
    sidebarPanel = shiny::sidebarPanel(
      width = 3,
      tagList(
        uiOutput(ns("pickerUI")),
        shinyWidgets::actionBtn(
          inputId = ns("btn_selectgrp"),
          label = "Select Group",
          style = "material-flat",
          color = "primary",
          size = "xs"
        ),
        br(),
        shinyWidgets::awesomeCheckboxGroup(
          inputId = ns("labeler_chkbox_plotopts"),
          label = "",
          choices = c(
            "Show Anomalies",
            "Show Legend"
          ),
          status = "danger"
        ),
        radioButtons(ns("brush_direction"),
          "Brush direction", c("xy", "x*"),
          inline = TRUE
        ),
        shiny::tableOutput(outputId = ns("labeler_metatable"))
      )
    ),
    mainPanel = shiny::mainPanel(
      width = 9,
      tagList(
        shiny::uiOutput(ns("tsplot_ui"), inline = T),
        shiny::uiOutput(ns("tsplot_zoomed_ui"), inline = T),
        reactable::reactableOutput(ns("dt_selectedpoints"))
      )
    )
  )
)
```

```
# Server
arrow_df <- shiny::reactive({
  arrow::open_dataset(arrow_ds_loc)
})

grp_unique_list <- shiny::reactive({
  arrow_df() |>
  dplyr::distinct(grp) |>
  dplyr::arrange(grp) |>
  dplyr::pull()
})

output$pickerUI <- shiny::renderUI({
  shinyWidgets::pickerInput(
    inputId = ns("selected_grps"),
    label = "Select Group(s)",
    choices = grp_unique_list(),
    # selected = metadata$grp_selected,
    multiple = TRUE,
    options = list(
      "actions-box" = TRUE,
      "selected-text-format" = "count > 3"
    )
  )
})

output$tag_pickerUI <- shiny::renderUI({
  shinyWidgets::pickerInput(
    inputId = ns("selected_tags"),
    label = "Select Tag(s)",
    choices = tag_choices()$tags,
    # selected = metadata$grp_selected,
    multiple = TRUE,
    options = list(
      "actions-box" = TRUE,
      "selected-text-format" = "count > 3"
    )
  )
})

tag_choices <- shiny::reactiveVal()

filtered_data <- shiny::eventReactive(input$btn_selectgrp, {
  tags <- arrow_df() |>
  dplyr::distinct(tag) |>
  dplyr::arrange(tag) |>
  dplyr::pull()

  tag_df <- dplyr::tibble(
    tag = tags,
    tag_color = RColorBrewer::brewer.pal(length(tags), "Set1")
  )
  tag_choices(tag_df)
})

arrow_df() |>
  dplyr::filter(grp != input$selected_grps) |>
  dplyr::collect() |>
  dplyr::arrange(grp, ds) |>
  dplyr::left_join(tag_df, by = "tag")
})

output$plot_ts <- shiny::renderPlot({
  dat <- filtered_data()
  par(mfrow = c(1, 2, M.F, N.F)) # (bottom, left, top, right)
  ts_plotter(
    dat = dat,
    plotopts = input$labeler_chkbox_plotopts,
    tag_choices_df = tag_choices()
  )
  res = 65
})

output$tsplot_ui <- shiny::renderUI({
  shiny::plotOutput(
    ns("plot_ts"),
    brush = brushOpts(
      id = ns("user_brush"),
      direction = input$brush_direction # "xy"
    ),
    dblclick = ns("user_dblclick"),
    height = "398px"
  )
})
```

```
})

output$tsplot_zoomed_ui <- shiny::renderUI({
  if (nrow(selectedPoints) == 0 | is.null(selectedPoints)) {
    return(NULL)
  }

  shiny::plotOutput(
    ns("plot_tszoomed"),
    brush = brushOpts(
      id = ns("user_brush_zoomed"),
      direction = input$brush_direction
    ),
    dblclick = ns("user_dblclick_zoomed"),
    height = "398px"
  )
})

selectedPoints <- shiny::reactive({
  shiny::brushedPoints(
    df = filtered_data(),
    brush = input$user_brush,
    xvar = "ds",
    yvar = "value"
  )
})

selectedPoints_zoomed <- shiny::reactive({
  shiny::brushedPoints(
    df = selectedPoints(),
    brush = input$user_brush_zoomed,
    xvar = "ds",
    yvar = "value"
  )
})

output$plot_tszoomed <- shiny::renderPlot({
  shiny::resample(selectedPoints())
  par(mfrow = c(1, 2, M.F, N.F)) # (bottom, left, top, right)
  ts_plotter(
    dat = selectedPoints(),
    plotopts = input$labeler_chkbox_plotopts,
    tag_choices_df = tag_choices()
  )
  res = 65
})

output$labeler_metatable <- shiny::renderTable({
  shiny::resample(filtered_data())
  shiny::resample(grp_unique_list())
  tibble::tibble(
    Parameter = c(
      "X Groups",
      "X Pts Above",
      "X Pts Below"
    ),
    Value = c(
      sprintf("%s/%s", length(input$selected_grps), length(grp_unique_list())),
      scales::label_cmmid()(nrow(filtered_data())),
      scales::label_cmmid()(nrow(selectedPoints()))
    )
  )
  spacing = "20",
  colnames = FALSE,
  bordered = FALSE
})

output$dt_selectedpoints <- reactable::renderReactable({
  dat <- selectedPoints_zoomed()

  if (nrow(dat) == 0 | is.null(dat)) {
    return(NULL)
  }

  reactable::reactable(
    dat,
    compact = TRUE,
    searchable = FALSE,
    filterable = TRUE,
    bordered = TRUE,
    defaultPageSize = 5,
    columns = list(
      ds = reactable::colDef(
        name = "Date",
        format = reactable::colFormat(
          date = TRUE, time = TRUE
        )
      ),
      value = reactable::colDef(
        name = "Value",
        format = reactable::colFormat(digits = 2, separators = TRUE)
      )
    )
  )
})
```

server.R