Interactive visualizations of large datasets using R, {shiny} and {arrow}

uRos 2024

Rahul Sangole, Nov 29, 2024

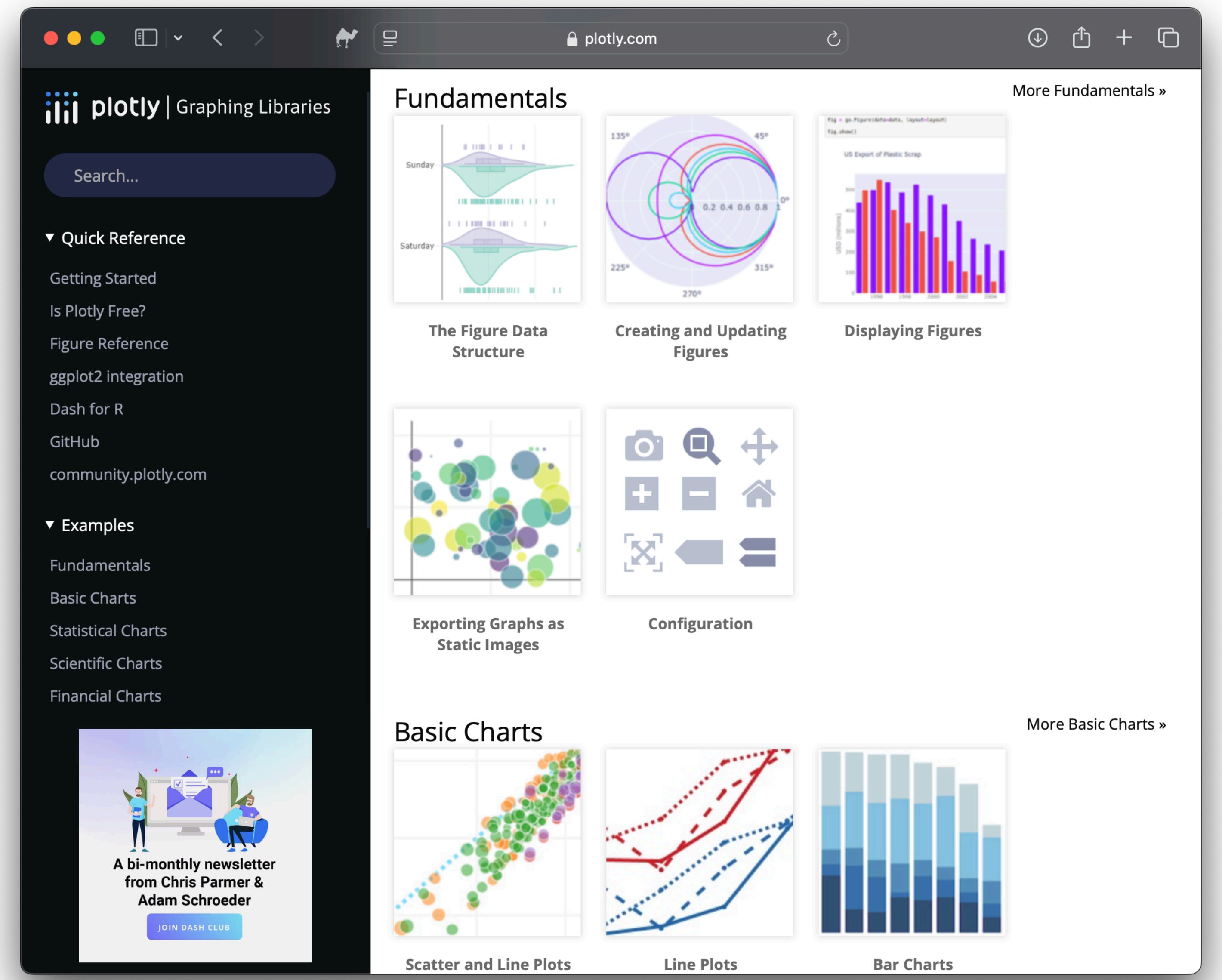
https://github.com/rsangole/uros2024-interactive_baseplots

/hello

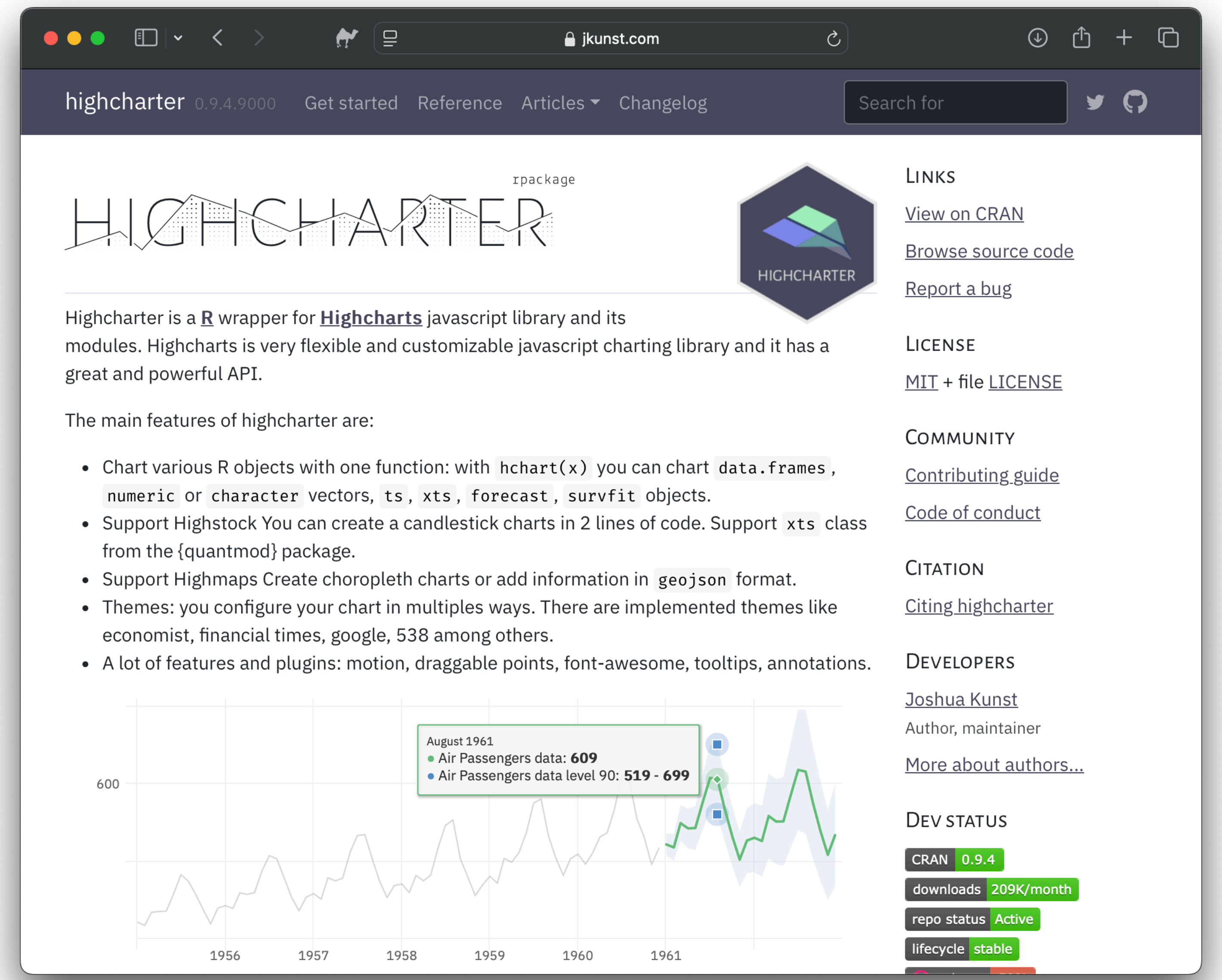
- Data scientist and manager, San Francisco
- Focus - Time series analyses, R in Production, Visualization, ML Ops

/motivation

- What are some interactive graphing options?



https://plotly.com/r/



Highcharter is a R wrapper for Highcharts javascript library and its modules. Highcharts is very flexible and customizable javascript charting library and it has a great and powerful API.

The main features of highcharter are:

- Chart various R objects with one function: with `hchart(x)` you can chart `data.frames`, `numeric` or `character` vectors, `ts`, `xts`, `forecast`, `survfit` objects.
- Support Highstock You can create a candlestick charts in 2 lines of code. Support `xts` class from the {quantmod} package.
- Support Highmaps Create choropleth charts or add information in `geojson` format.
- Themes: you configure your chart in multiples ways. There are implemented themes like economist, financial times, google, 538 among others.
- A lot of features and plugins: motion, draggable points, font-awesome, tooltips, annotations.

LINKS
View on CRAN
Browse source code
Report a bug

LICENSE
MIT + file LICENSE

COMMUNITY
Contributing guide
Code of conduct

CITATION
Citing highcharter

DEVELOPERS
Joshua Kunst
Author, maintainer
More about authors...

DEV STATUS
CRAN 0.9.4
downloads 209K/month
repo status Active
lifecycle stable

https://jkunst.com/highcharter/

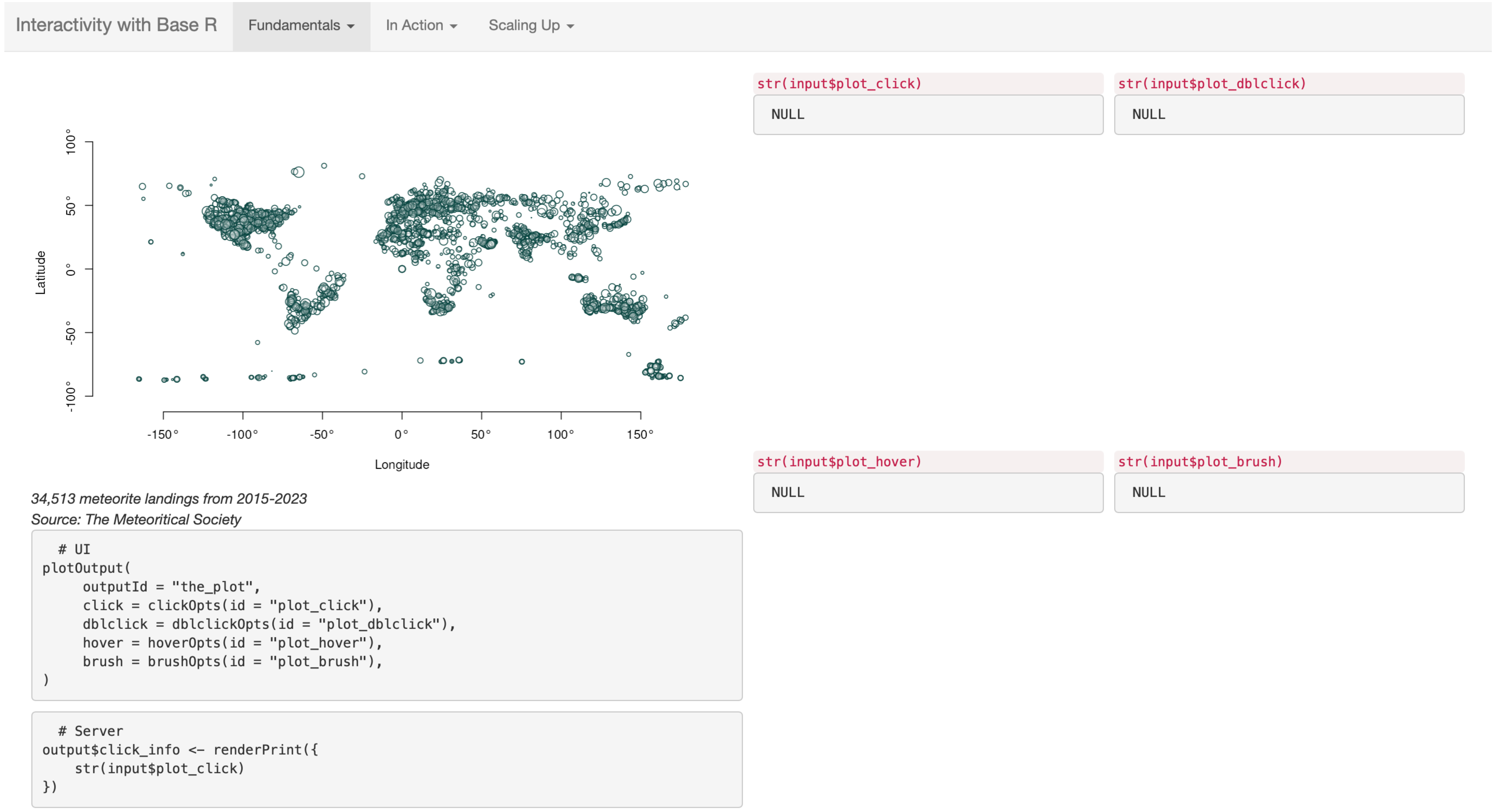
- What are our challenges?

Handling large data

Visualizing large data

/demo

Basic shiny interactivity



34,513 meteorite landings from 2015-2023
Source: The Meteoritical Society

```
 # UI
plotOutput(
     outputId = "the_plot",
     click = clickOpts(id = "plot_click"),
     dblclick = dblclickOpts(id = "plot_dblclick"),
     hover = hoverOpts(id = "plot_hover"),
     brush = brushOpts(id = "plot_brush"),
)
```

```
 # Server
output$click_info <- renderPrint({
    str(input$plot_click)
})
```

`str(input$plot_click)`

```
NULL
```

`str(input$plot_dblclick)`

```
NULL
```

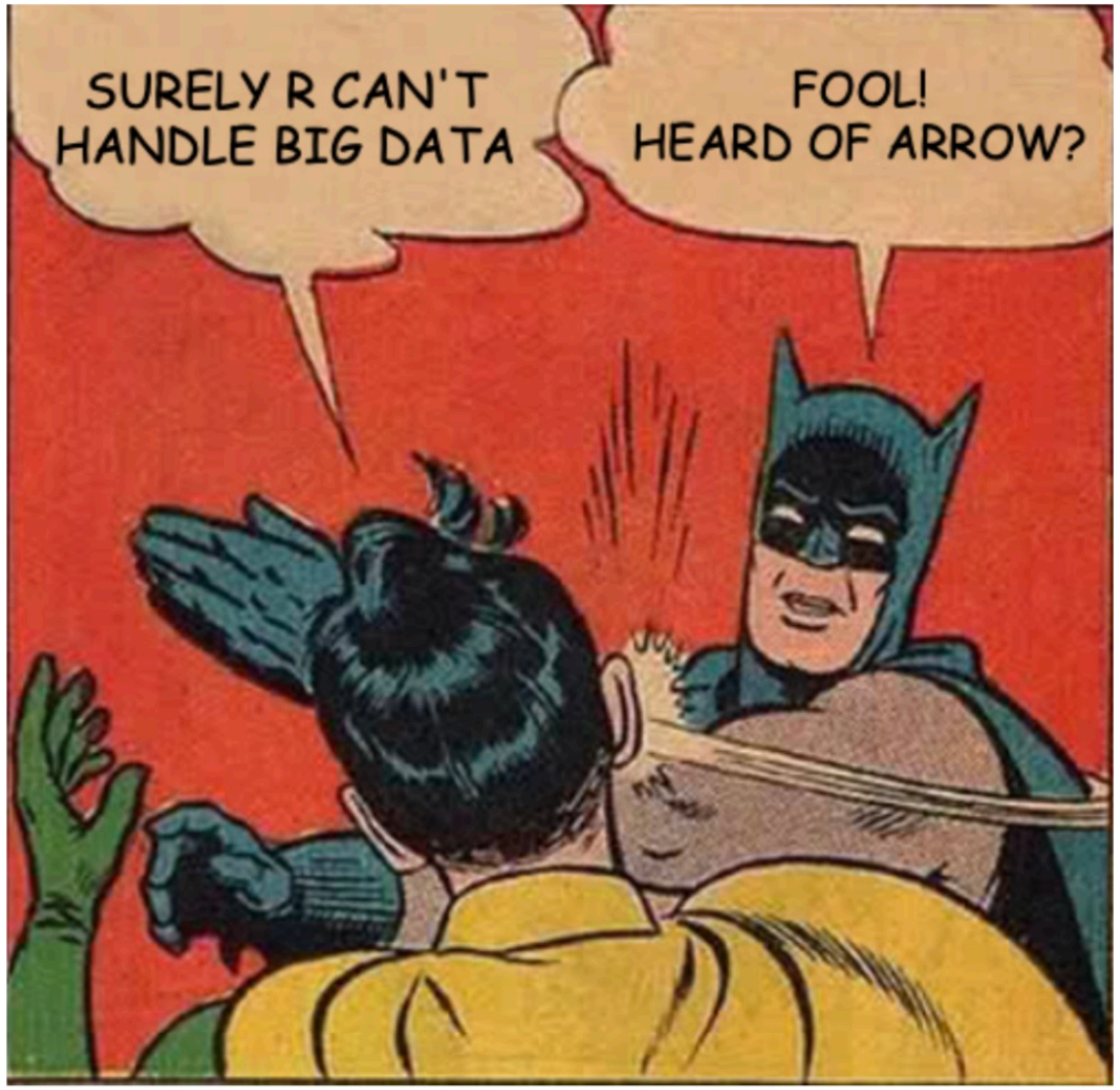
`str(input$plot_hover)`

```
NULL
```

`str(input$plot_brush)`

```
NULL
```

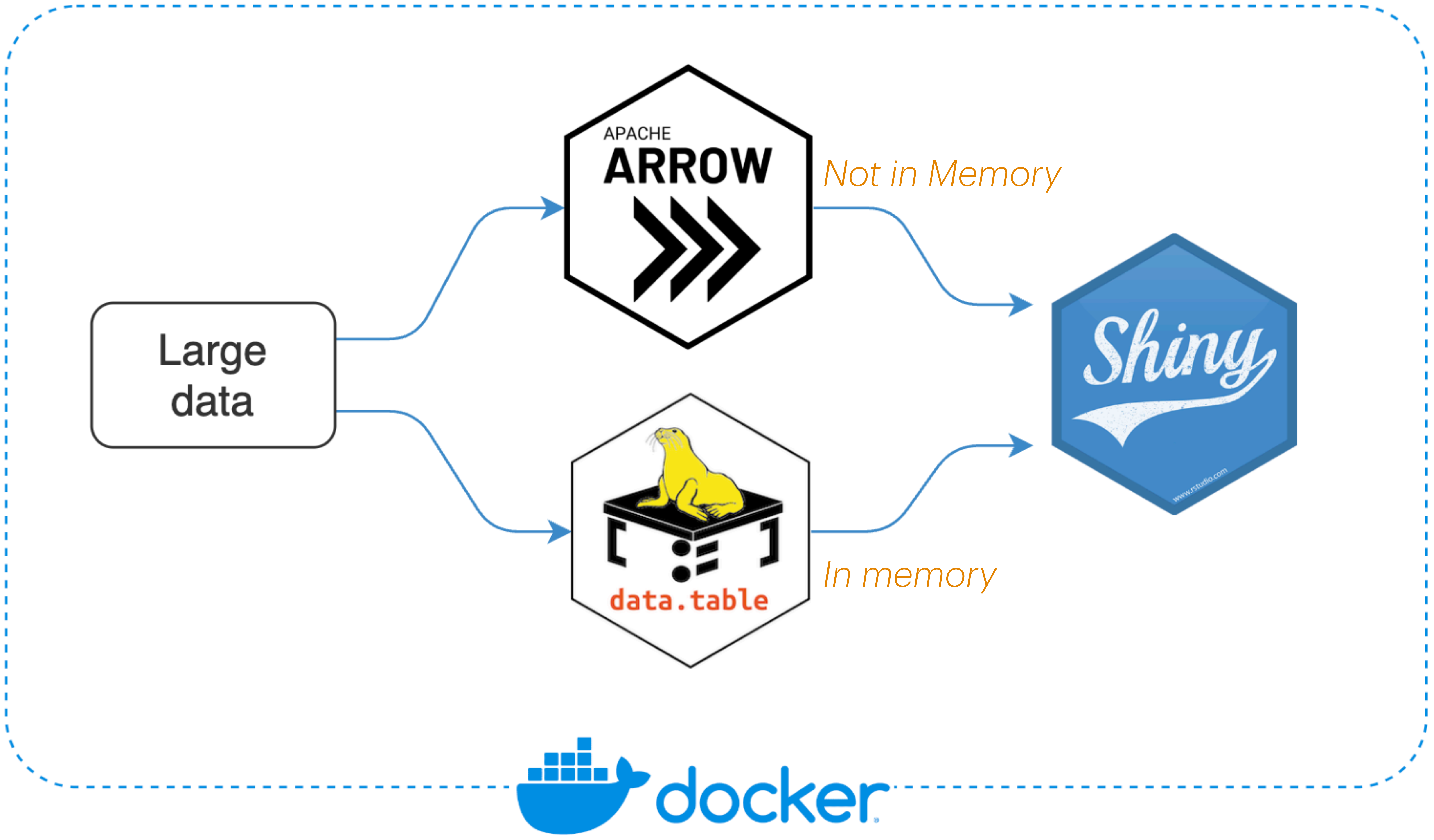
/working with larger datasets



** How large is "large"?*

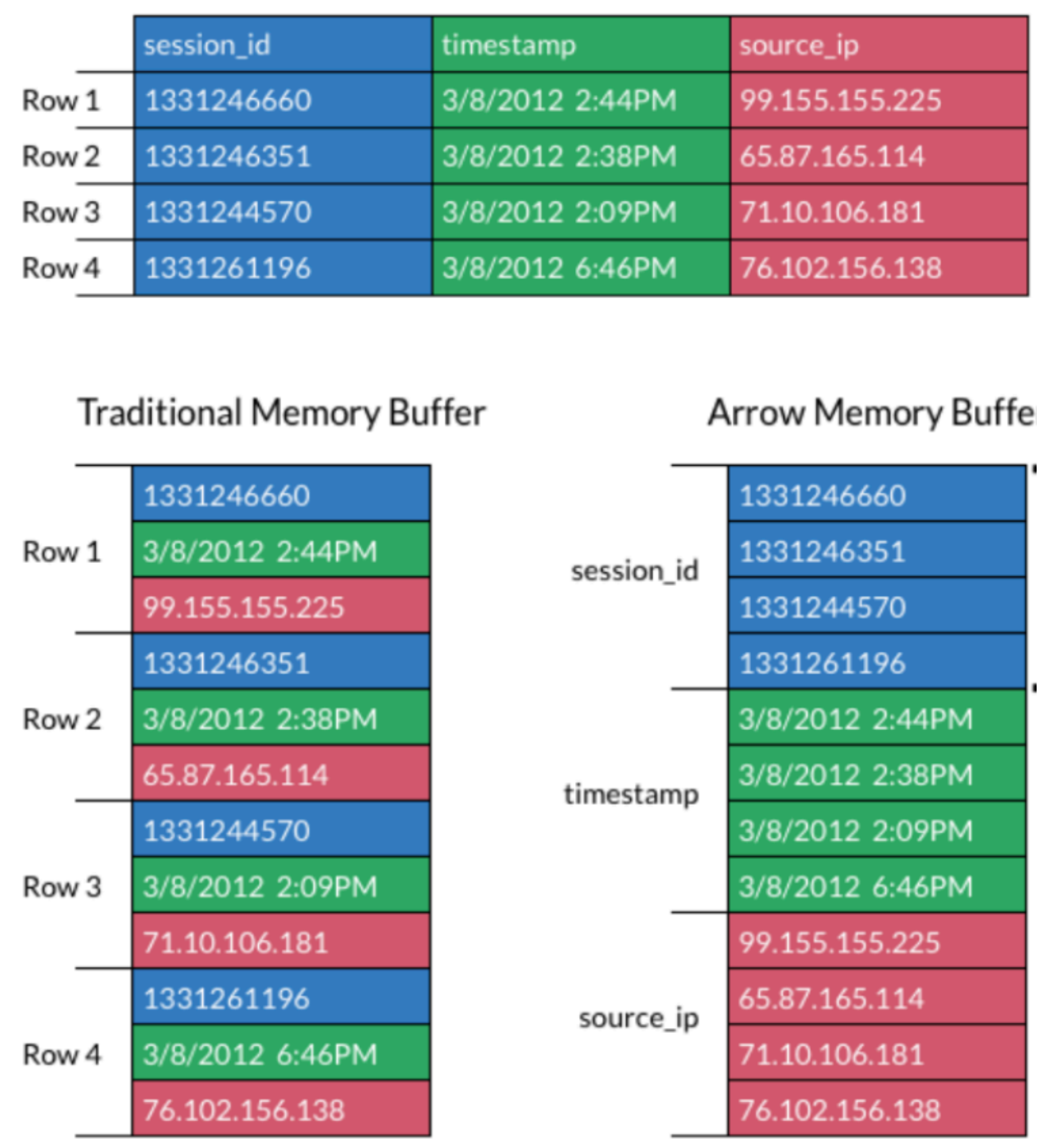
/working with larger datasets

two-approaches



/arrow + parquet

Parquet



	session_id	timestamp	source_ip
Row 1	1331246660	3/8/2012 2:44PM	99.155.155.225
Row 2	1331246351	3/8/2012 2:38PM	65.87.165.114
Row 3	1331244570	3/8/2012 2:09PM	71.10.106.181
Row 4	1331261196	3/8/2012 6:46PM	76.102.156.138

Traditional Memory Buffer

```
Row 1  1331246660
       3/8/2012 2:44PM
       99.155.155.225
Row 2  1331246351
       3/8/2012 2:38PM
       65.87.165.114
Row 3  1331244570
       3/8/2012 2:09PM
       71.10.106.181
Row 4  1331261196
       3/8/2012 6:46PM
       76.102.156.138
```

Arrow Memory Buffe

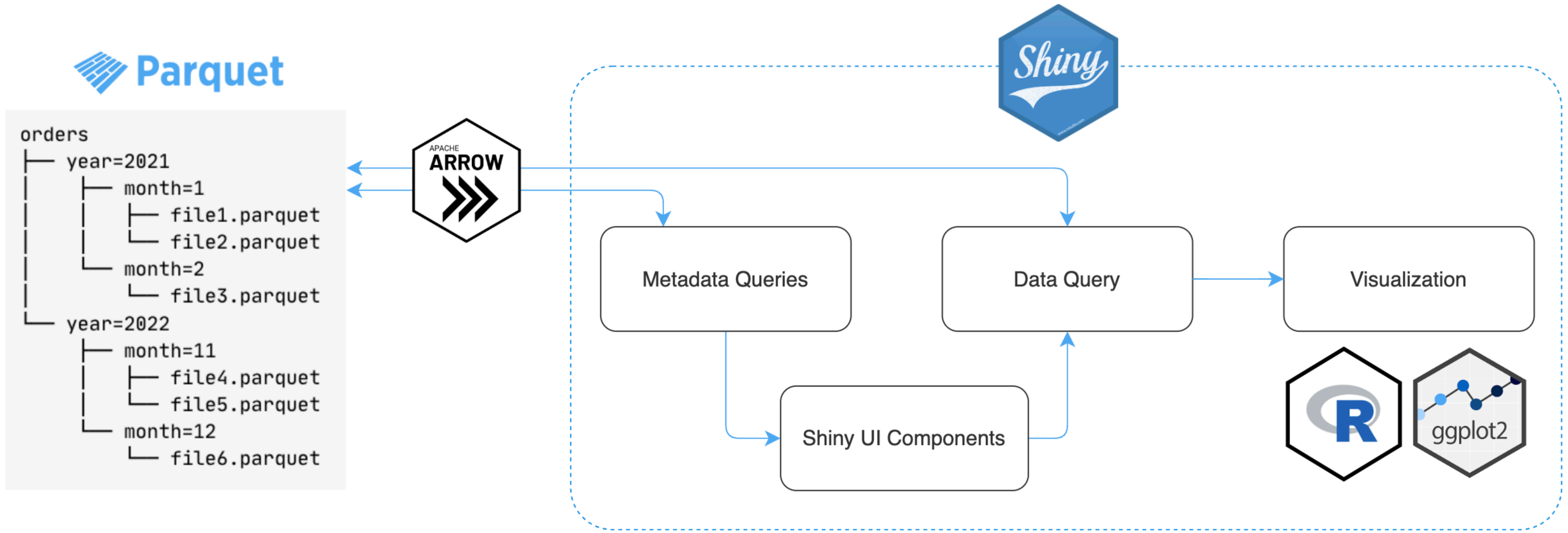
```
session_id  1331246660
            1331246351
            1331244570
            1331261196
timestamp   3/8/2012 2:44PM
            3/8/2012 2:38PM
            3/8/2012 2:09PM
            3/8/2012 6:46PM
source_ip   99.155.155.225
            65.87.165.114
            71.10.106.181
            76.102.156.138
```

APACHE ARROW

- `{arrow}` evaluates lazily by default
- Verbs: `filter`, `select`, `mutate`, `join`, `distinct`, `group_by` + summarize, and across
- execution only runs on `dplyr::collect()`
- massive performance gains using parquet files and smart partitioning

dplyr

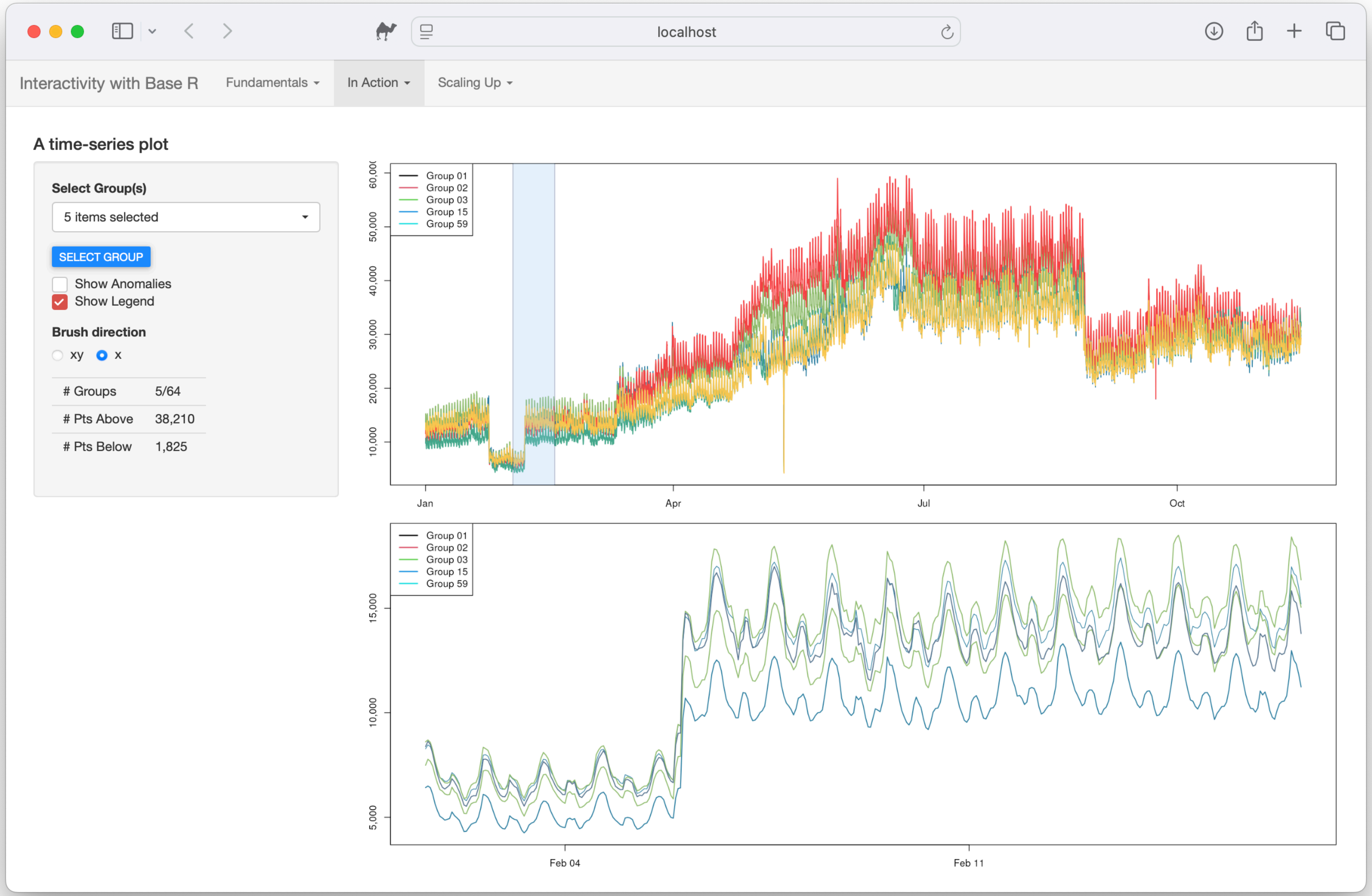
/shiny architecture



```
orders
├── year=2021
│   ├── month=1
│   │   ├── file1.parquet
│   │   └── file2.parquet
│   └── month=2
│       └── file3.parquet
└── year=2022
    ├── month=11
    │   ├── file4.parquet
    │   └── file5.parquet
    └── month=12
        └── file6.parquet
```

/demo

Large data with arrow



A time-series plot

Select Group(s)

5 items selected

- [ ] Show Anomalies
- [x] Show Legend

Brush direction

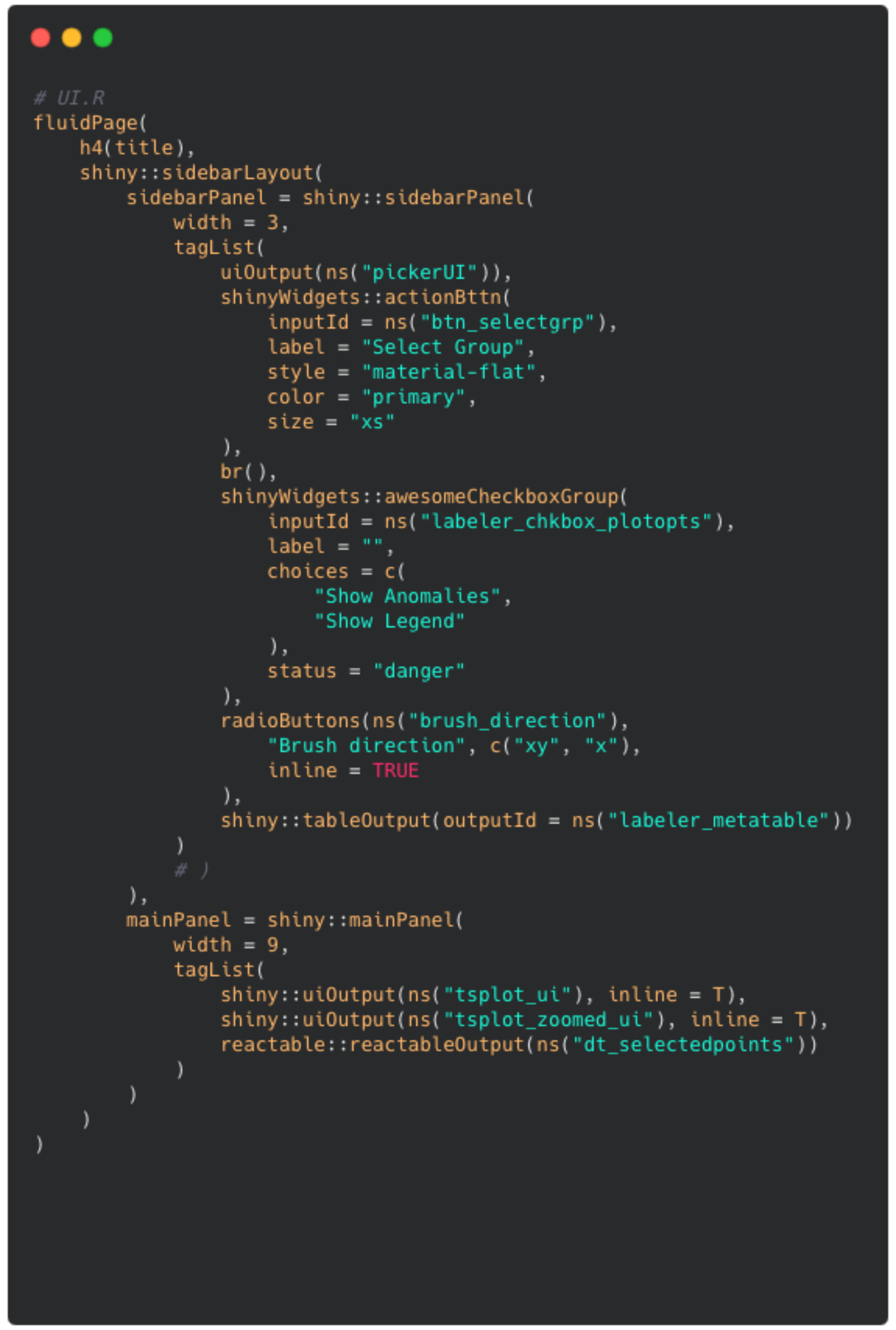
xy [✓ x]

# Groups	5/64
# Pts Above	38,210
# Pts Below	1,825

/scalability

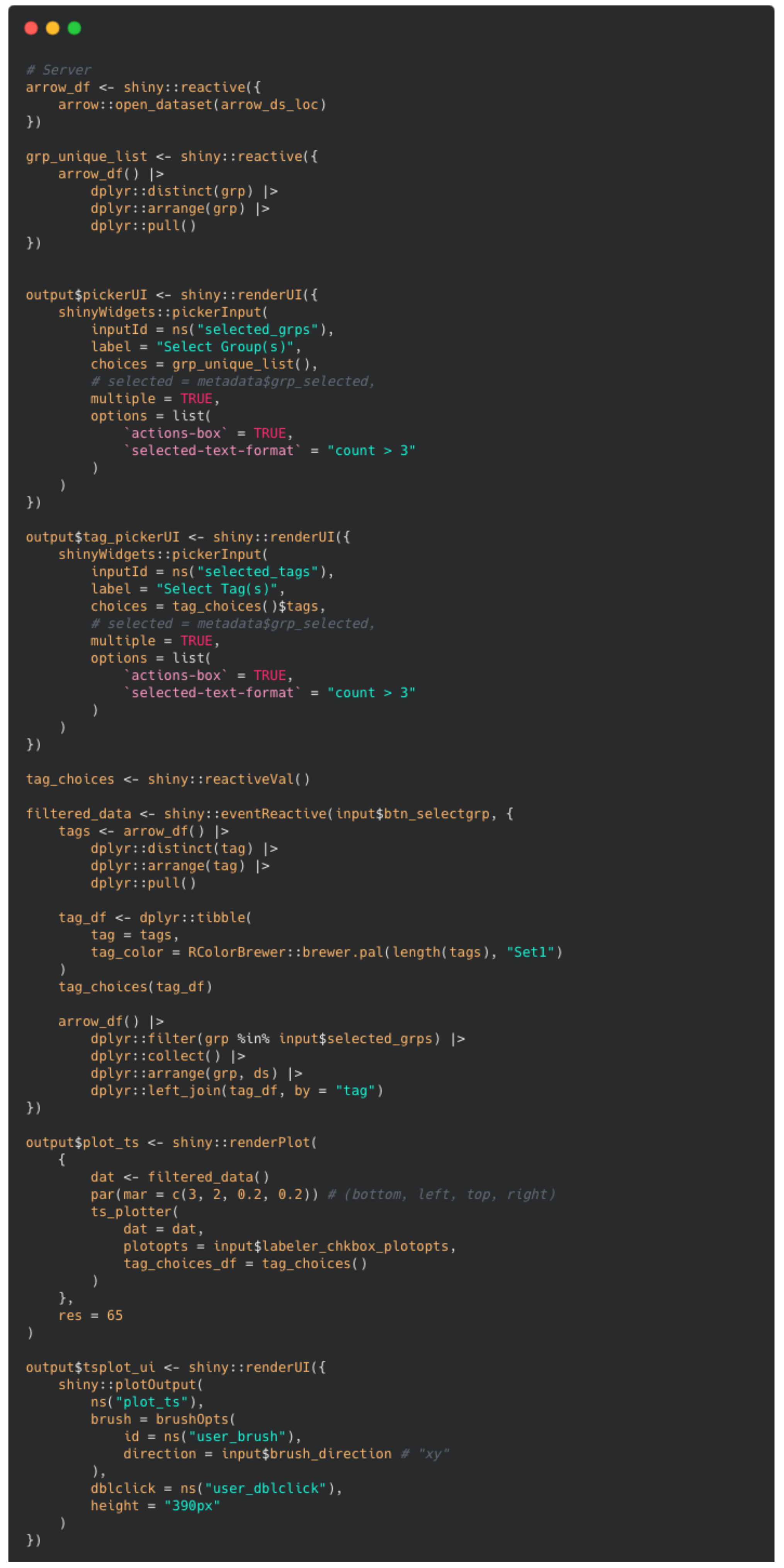
The challenge with simple Shiny

ui.R

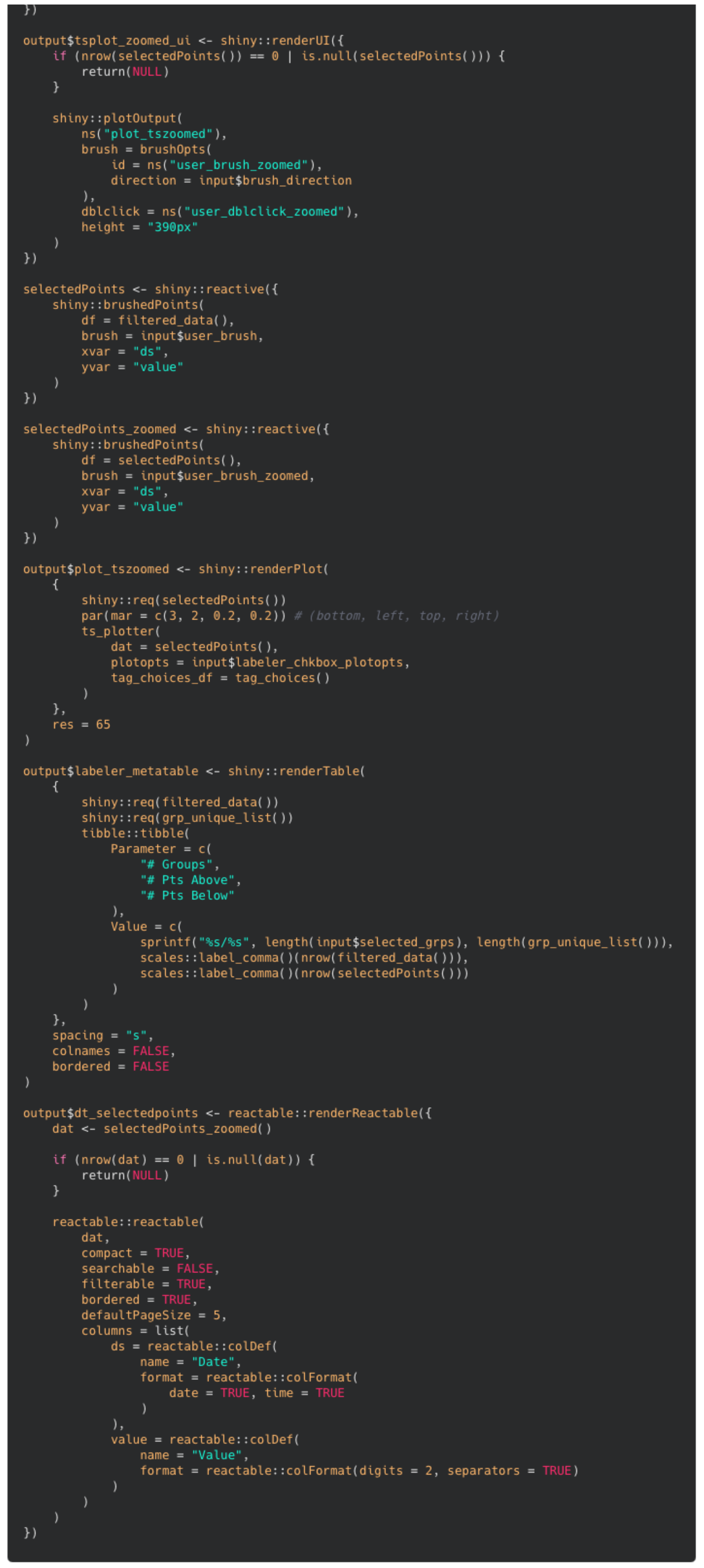


```r
# UI.R
fluidPage(
    h4(title),
    shiny::sidebarLayout(
        sidebarPanel = shiny::sidebarPanel(
            width = 3,
            tagList(
                uiOutput(ns("pickerUI")),
                shinyWidgets::actionBttn(
                    inputId = ns("btn_selectgrp"),
                    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.R



```r
# 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 %in% input$selected_grps) |>
        dplyr::collect() |>
        dplyr::arrange(grp, ds) |>
        dplyr::left_join(tag_df, by = "tag")
})

output$plot_ts <- shiny::renderPlot(
    {
        dat <- filtered_data()
        par(mar = c(3, 2, 0.2, 0.2)) # (bottom, left, top, right)
        ts_plotter(
            dat = dat,
            plotopts = input$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 = "390px"
    )
})
```



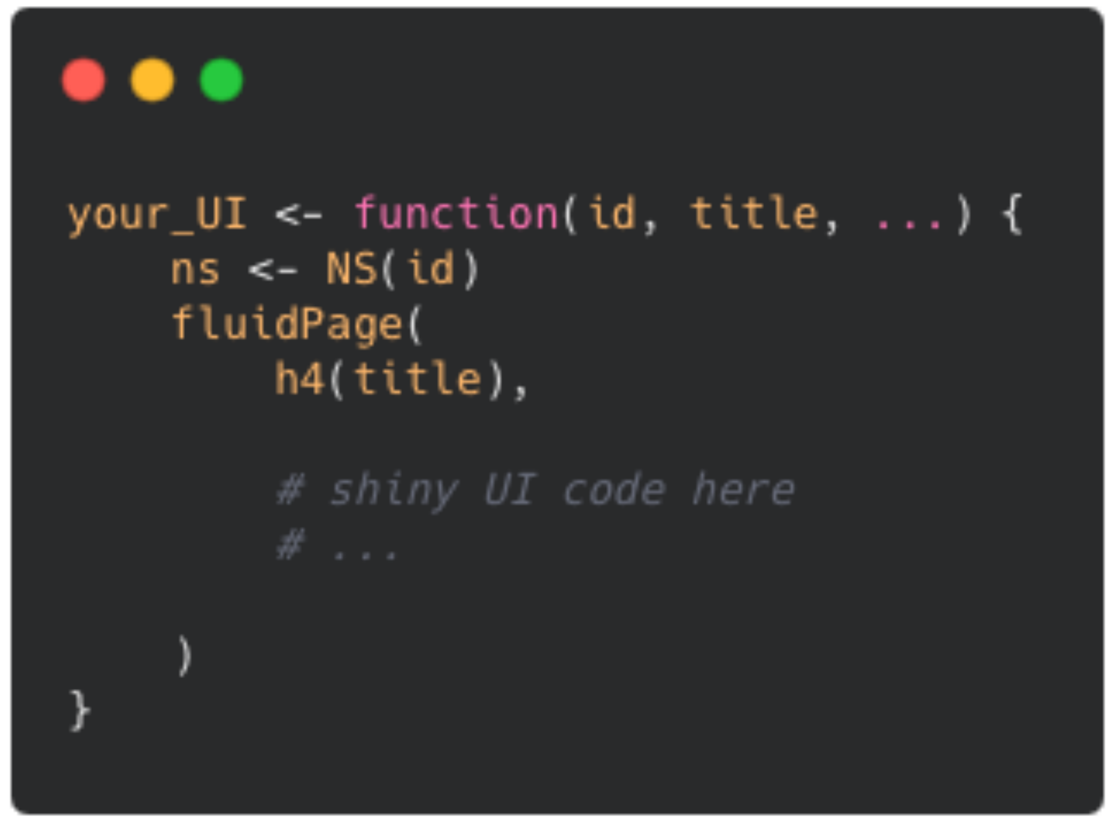
```r
})

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 = "390px"
    )
})

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$plot_tszoomed <- shiny::renderPlot(
    {
        shiny::req(selectedPoints())
        par(mar = c(3, 2, 0.2, 0.2)) # (bottom, left, top, right)
        ts_plotter(
            dat = selectedPoints(),
            plotopts = input$labeler_chkbox_plotopts,
            tag_choices_df = tag_choices()
        )
    },
    res = 65
)

output$labeler_metatable <- shiny::renderTable(
    {
        shiny::req(filtered_data())
        shiny::req(grp_unique_list())
        tibble::tibble(
            Parameter = c(
                "# Groups",
                "# Pts Above",
                "# Pts Below"
            ),
            Value = c(
                sprintf("%s/%s", length(input$selected_grps), length(grp_unique_list())),
                scales::label_comma()(nrow(filtered_data())),
                scales::label_comma()(nrow(selectedPoints()))
            )
        )
    },
    spacing = "s",
    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)
            )
        )
    )
})
```

/shiny-modules

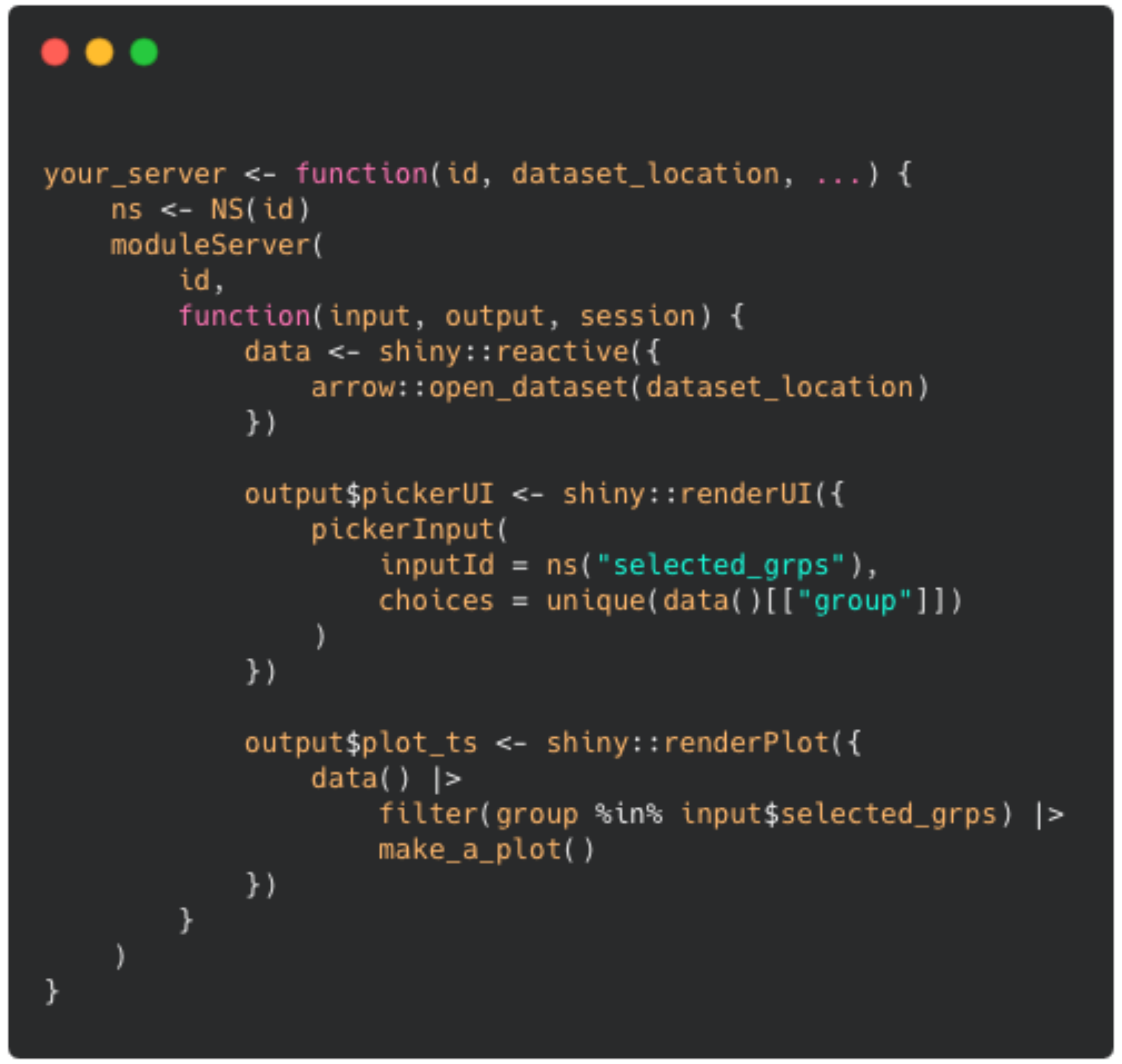
The solution to scalability

- Modules are *functions* they help you reuse code;
  anything you can do with a function, you can do with a module

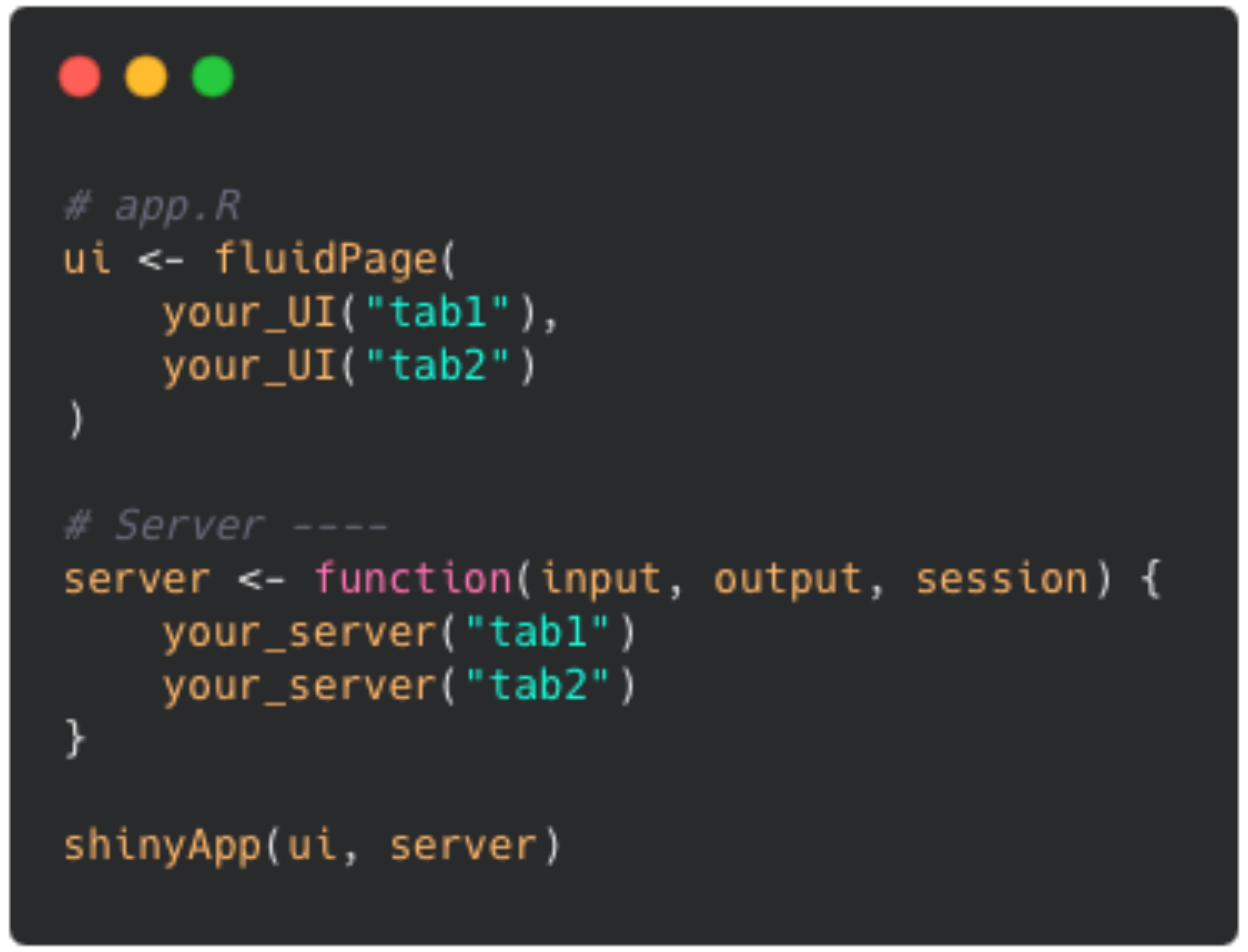
- *Namespacing makes it easier*



```r
your_UI <- function(id, title, ...) {
    ns <- NS(id)
    fluidPage(
        h4(title),

        # shiny UI code here
        # ...

    )
}
```



```r
your_server <- function(id, dataset_location, ...) {
    ns <- NS(id)
    moduleServer(
        id,
        function(input, output, session) {
            data <- shiny::reactive({
                arrow::open_dataset(dataset_location)
            })

            output$pickerUI <- shiny::renderUI({
                pickerInput(
                    inputId = ns("selected_grps"),
                    choices = unique(data()[["group"]])
                )
            })

            output$plot_ts <- shiny::renderPlot({
                data() |>
                    filter(group %in% input$selected_grps) |>
                    make_a_plot()
            })
        }
    )
}
```



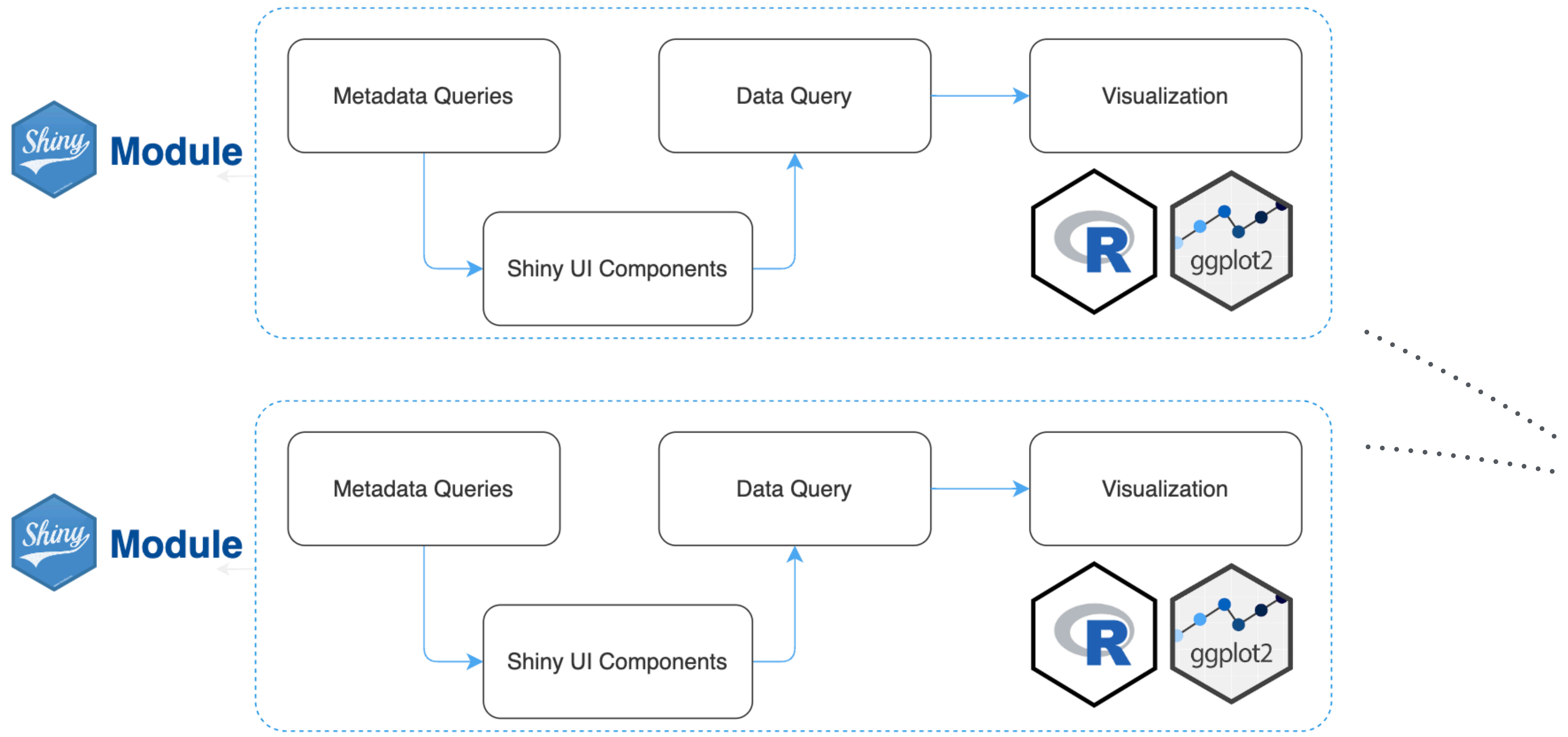
```r
# app.R
ui <- fluidPage(
    your_UI("tab1"),
    your_UI("tab2")
)

# Server ----
server <- function(input, output, session) {
    your_server("tab1")
    your_server("tab2")
}

shinyApp(ui, server)
```

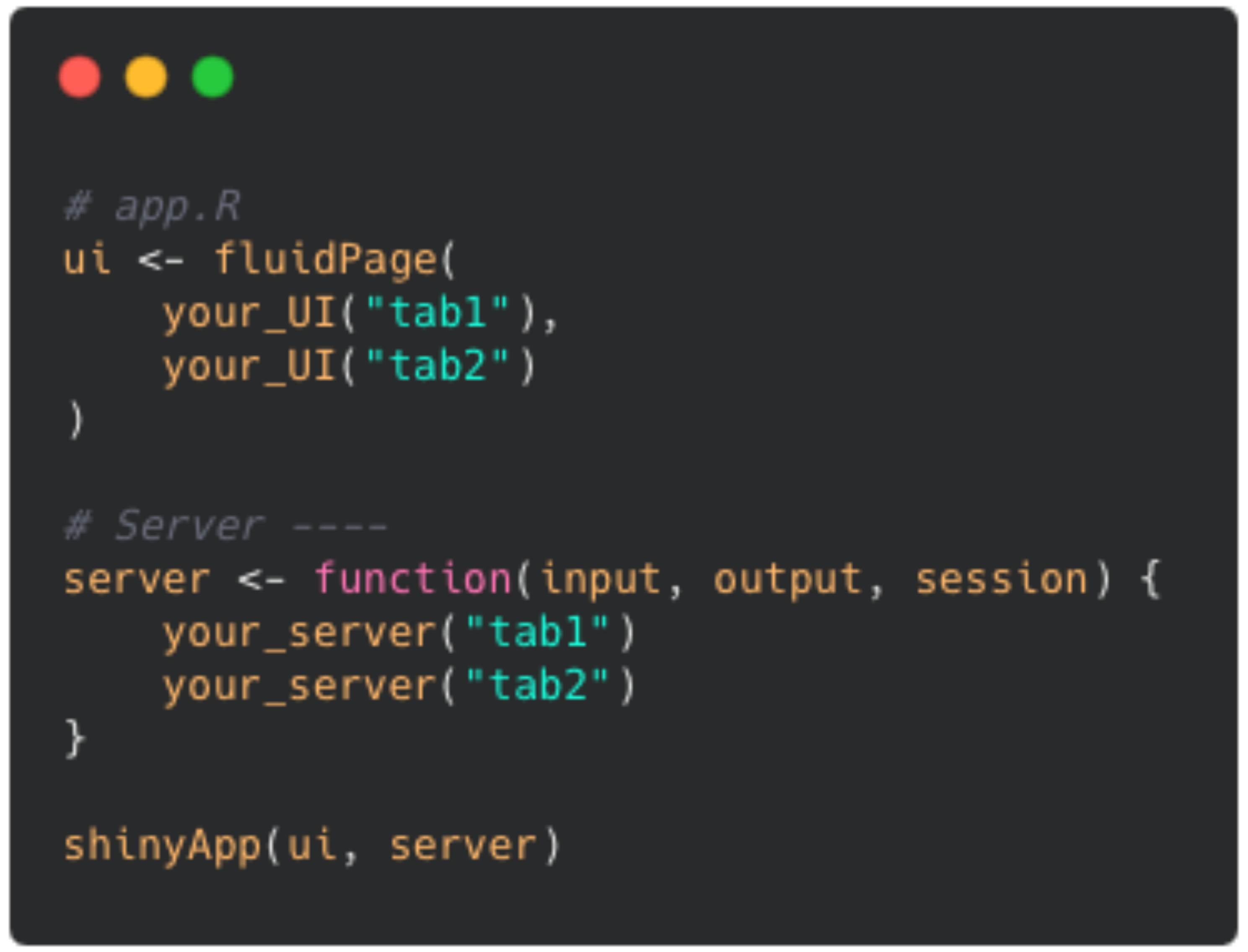
/shiny-modules

The solution to scalability

- Modules are *functions* they help you reuse code;
  anything you can do with a function, you can do with a module

- *Namespacing makes it easier*

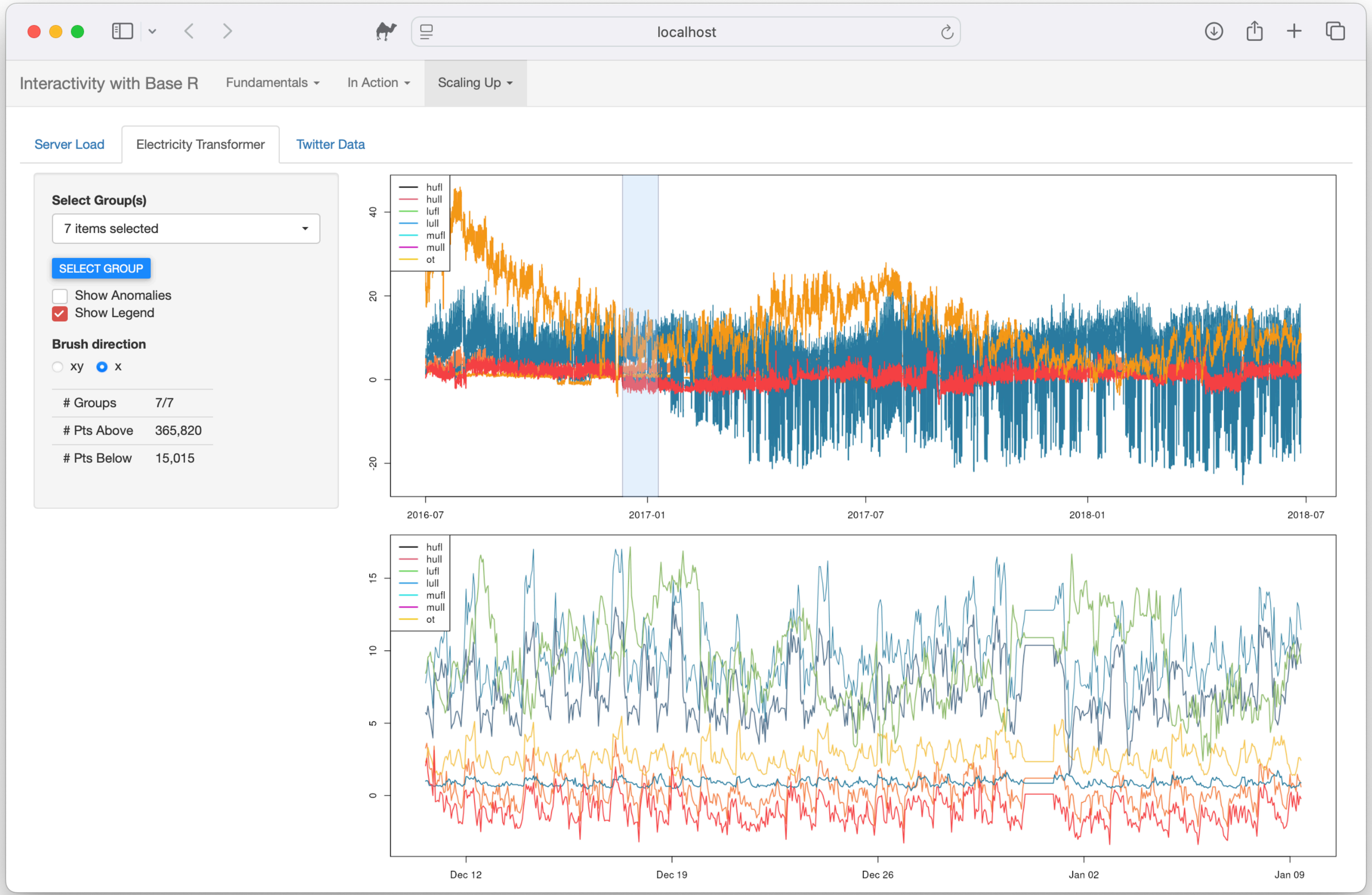




```r
# app.R
ui <- fluidPage(
    your_UI("tab1"),
    your_UI("tab2")
)

# Server ----
server <- function(input, output, session) {
    your_server("tab1")
    your_server("tab2")
}

shinyApp(ui, server)
```

/demo

Shiny modules



Select Group(s)

7 items selected

- [ ] Show Anomalies
- [x] Show Legend

Brush direction

xy [✓ x]

# Groups	7/7
# Pts Above	365,820
# Pts Below	15,015

/in conclusion

- R really has something special…
  - Lazy-evaluation using {arrow}
  - Columnar, partitioned storage using {parquet}
  - Speedy execution, capability of the base R & {ggplot} plotting packages
  - Interactivity using Shiny
  - Scalability using shiny modules

/thanks

rsangole

rahulsangole