Taking Flight with Shiny: A Modules-First Learning Approach

Emily Riederer

emilyriederer.com || @emilyriederer

To learn to fly, you build end-to-end skills then scale

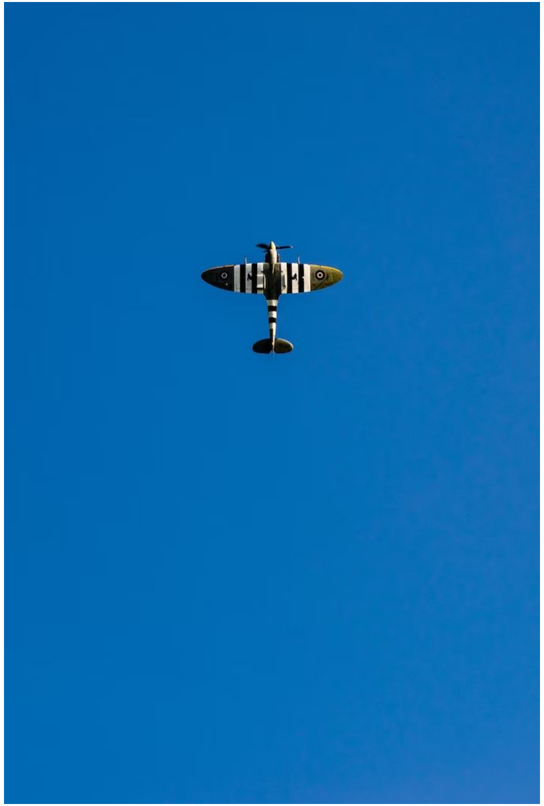


Photo Credit: Richard Schunemann on Unsplash

You should...

✅ Successfully play in a flight simulator before

✅ Flying a small plane before

✅ Becoming an international commercial pilot

To learn to fly, you build end-to-end skills then scale

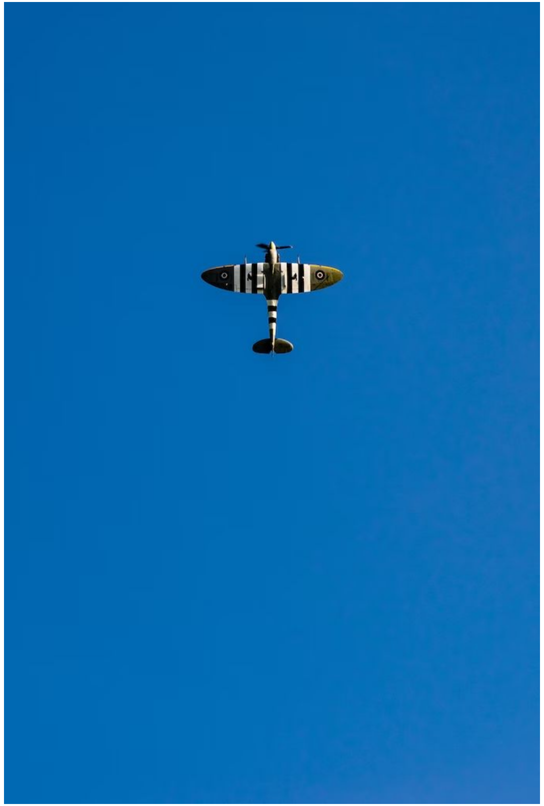


Photo Credit: Richard Schunemann on Unsplash

You should...

✅ Successfully play in a flight simulator before

✅ Flying a small plane before

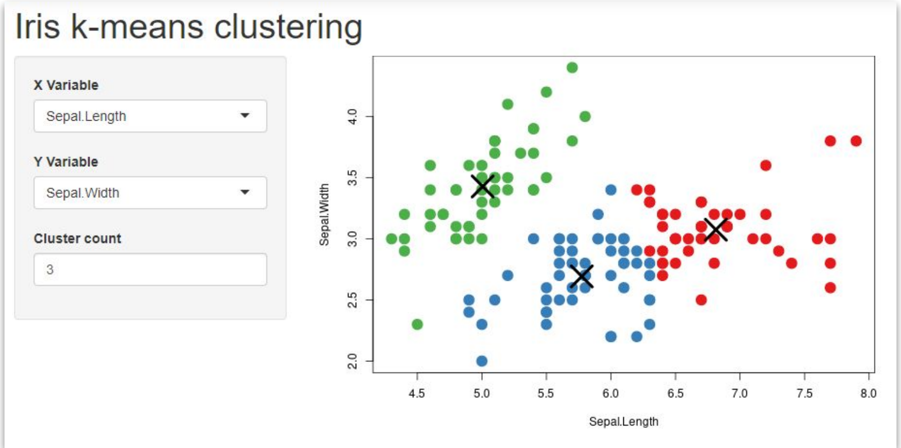
✅ Becoming an international commercial pilot

You wouldn't...

❌ Start learning on the largest system available

❌ Test your 'take-off' skill on a Boeing before learning to land

Shiny is taught as a monolith that's hard to read and debug



Iris k-means clustering

X Variable

Sepal.Length

Y Variable

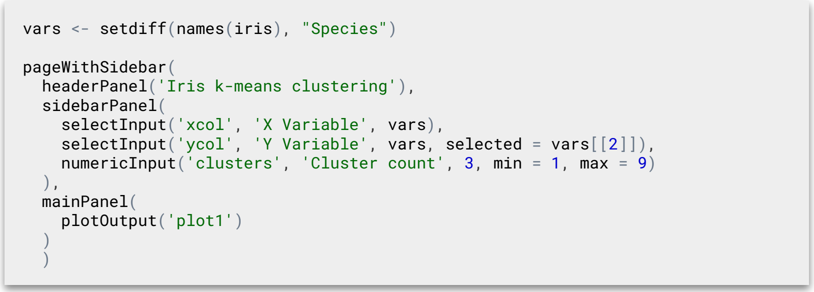
Sepal.Width

Cluster count

3

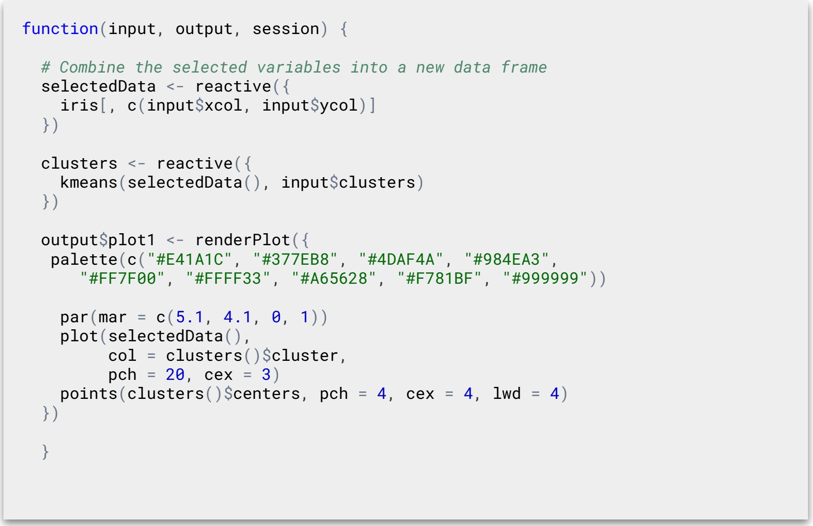
Source: Posit's Shiny Gallery

ui.R



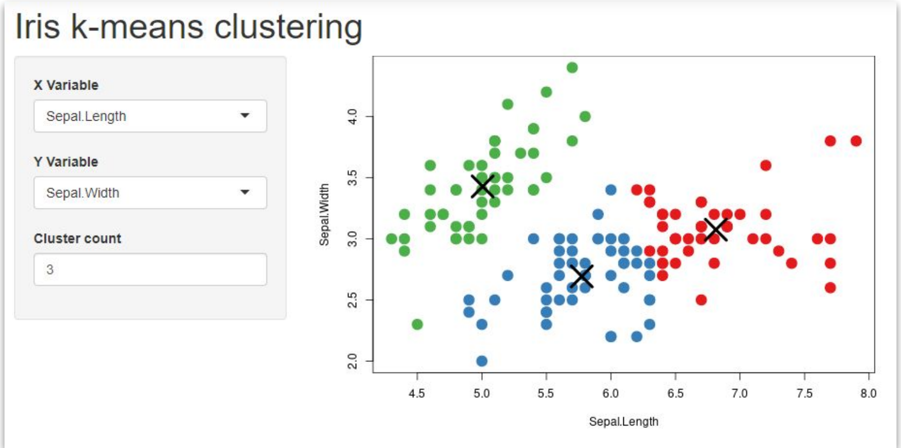
```
vars <- setdiff(names(iris), "Species")

pageWithSidebar(
  headerPanel('Iris k-means clustering'),
  sidebarPanel(
    selectInput('xcol', 'X Variable', vars),
    selectInput('ycol', 'Y Variable', vars, selected = vars[[2]]),
    numericInput('clusters', 'Cluster count', 3, min = 1, max = 9)
  ),
  mainPanel(
    plotOutput('plot1')
  )
  )
```

server.R



```
function(input, output, session) {

  # Combine the selected variables into a new data frame
  selectedData <- reactive({
    iris[, c(input$xcol, input$ycol)]
  })

  clusters <- reactive({
    kmeans(selectedData(), input$clusters)
  })

  output$plot1 <- renderPlot({
   palette(c("#E41A1C", "#377EB8", "#4DAF4A", "#984EA3",
      "#FF7F00", "#FFFF33", "#A65628", "#F781BF", "#999999"))

    par(mar = c(5.1, 4.1, 0, 1))
    plot(selectedData(),
         col = clusters()$cluster,
         pch = 20, cex = 3)
    points(clusters()$centers, pch = 4, cex = 4, lwd = 4)
  })

  }
```

Shiny is taught as a monolith that’s hard to read and debug



Iris k-means clustering

X Variable

Sepal.Length

Y Variable

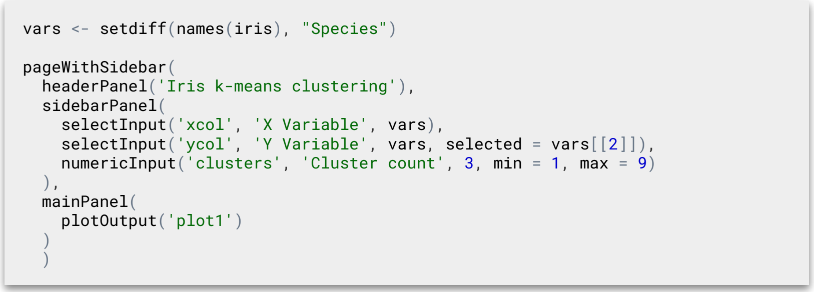
Sepal.Width

Cluster count

3

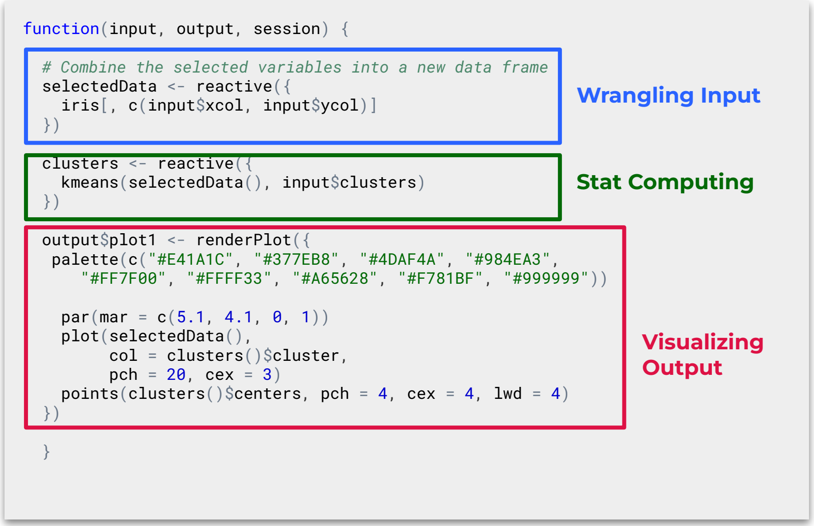
Source: Posit’s Shiny Gallery

ui.R



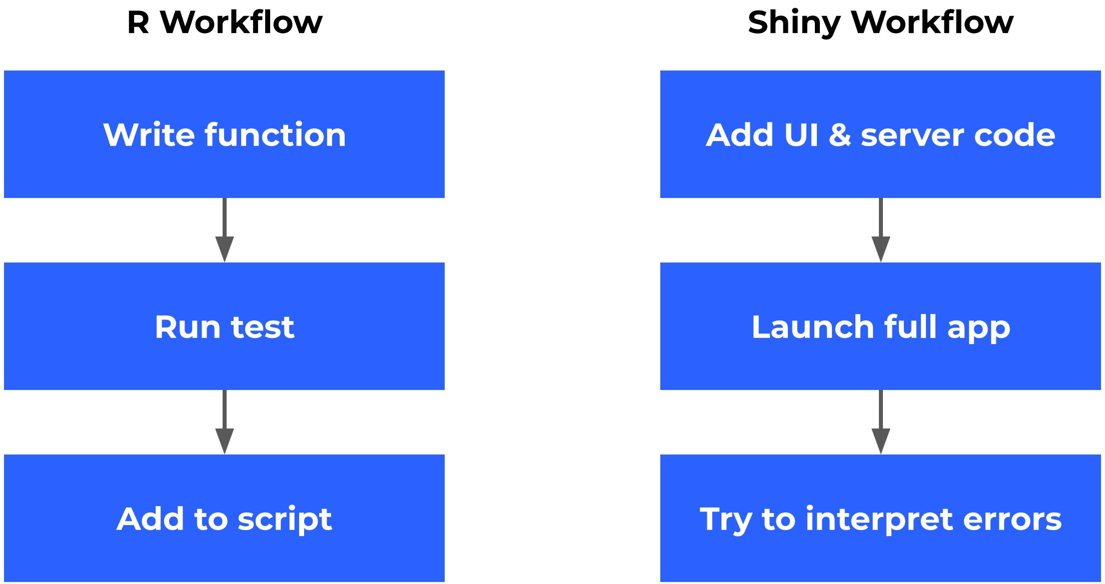
```r
vars <- setdiff(names(iris), "Species")

pageWithSidebar(
  headerPanel('Iris k-means clustering'),
  sidebarPanel(
    selectInput('xcol', 'X Variable', vars),
    selectInput('ycol', 'Y Variable', vars, selected = vars[[2]]),
    numericInput('clusters', 'Cluster count', 3, min = 1, max = 9)
  ),
  mainPanel(
    plotOutput('plot1')
  )
  )
```

server.R



```r
function(input, output, session) {

  # Combine the selected variables into a new data frame
  selectedData <- reactive({
    iris[, c(input$xcol, input$ycol)]
  })

  clusters <- reactive({
    kmeans(selectedData(), input$clusters)
  })

  output$plot1 <- renderPlot({
   palette(c("#E41A1C", "#377EB8", "#4DAF4A", "#984EA3",
      "#FF7F00", "#FFFF33", "#A65628", "#F781BF", "#999999"))

    par(mar = c(5.1, 4.1, 0, 1))
    plot(selectedData(),
         col = clusters()$cluster,
         pch = 20, cex = 3)
    points(clusters()$centers, pch = 4, cex = 4, lwd = 4)
  })

  }
```

This approach creates poor developer workflows and conflicts with how best-practice R development patterns



R Workflow

Write function

Run test

Add to script

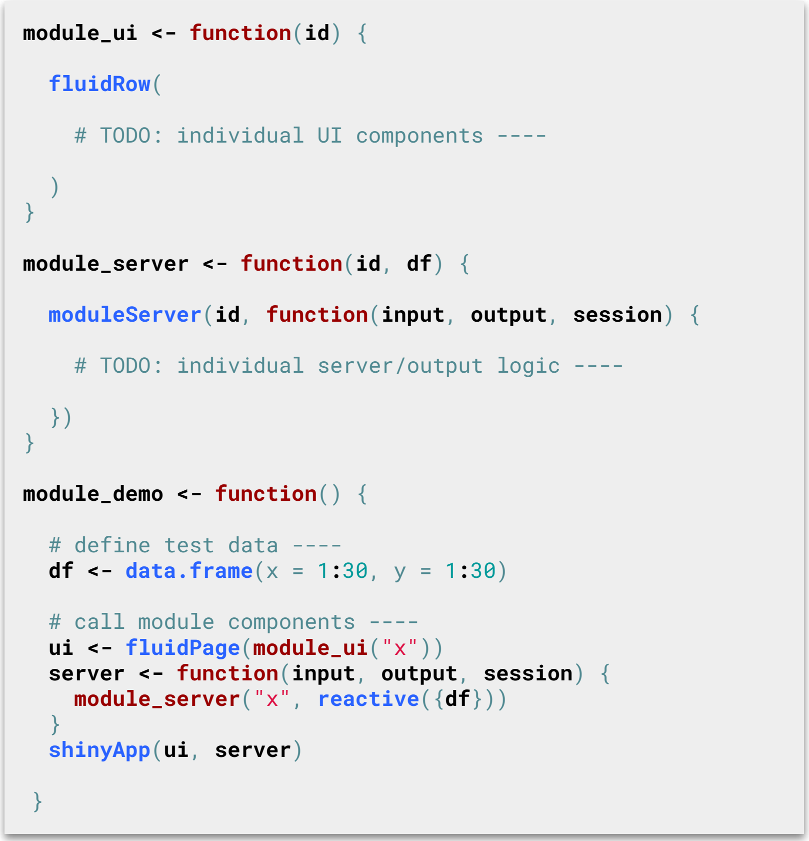
Shiny Workflow

Add UI & server code

Launch full app

Try to interpret errors

Modules offer the same separation-of-concerns as R functions



```r
module_ui <- function(id) {

  fluidRow(

    # TODO: individual UI components ----

  )
}

module_server <- function(id, df) {

  moduleServer(id, function(input, output, session) {

    # TODO: individual server/output logic ----

  })
}

module_demo <- function() {

  # define test data ----
  df <- data.frame(x = 1:30, y = 1:30)

  # call module components ----
  ui <- fluidPage(module_ui("x"))
  server <- function(input, output, session) {
    module_server("x", reactive({df}))
  }
  shinyApp(ui, server)

}
```

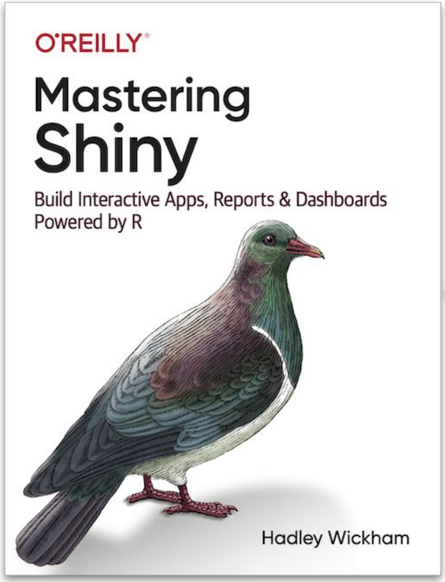
Modules offer the same separation-of-concerns as R functions

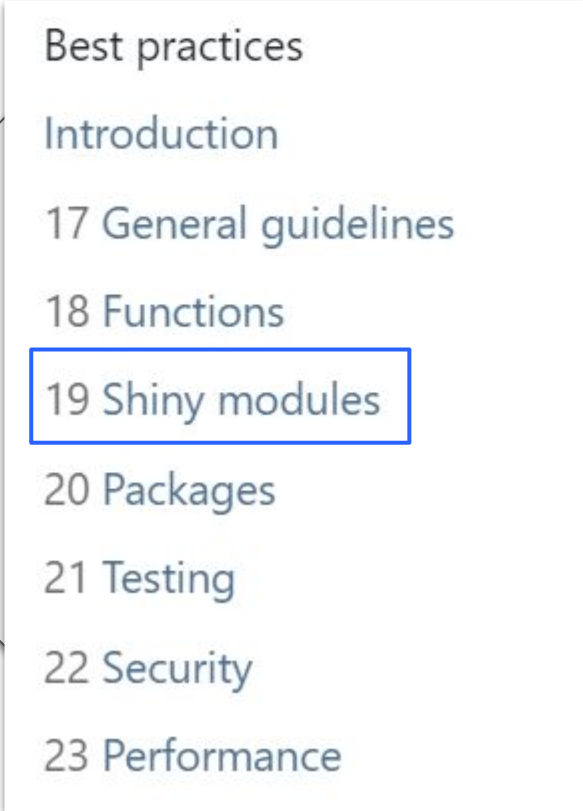
✅ Isolated functionality

✅ Independent testability

✅ Composability

We should learn and teach Shiny 'modules-first'




Best practices

Introduction

17 General guidelines

18 Functions

19 Shiny modules

20 Packages

21 Testing

22 Security

23 Performance

Case study: Flight delay dashboard

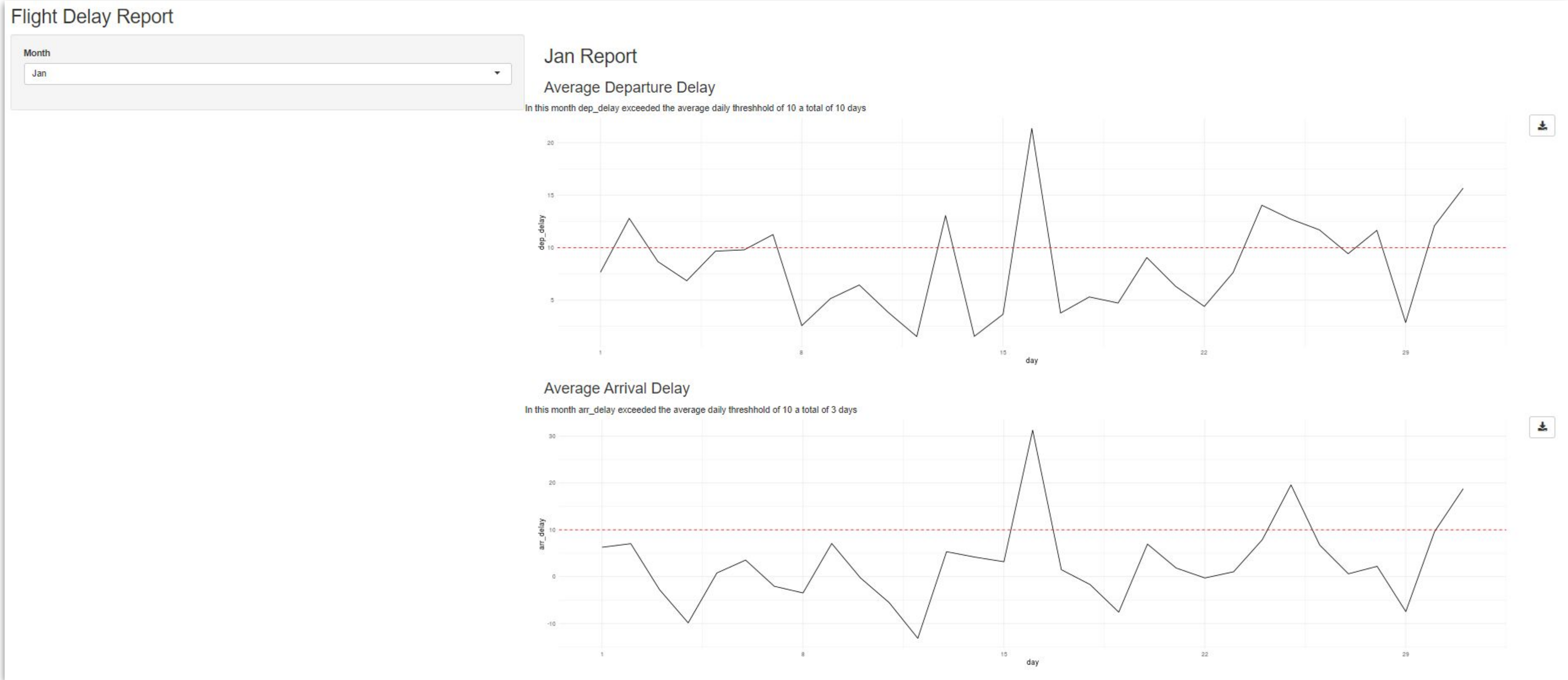
Photo Credit: Matthew Smith on Unsplash

Imagine we work at an airline on a team responsible for tracking on-time flight performance. We want to:

- Track multiple metrics
- Examine performance across time
- Evaluate success relative to a pre-defined goal
- Enable analysis, reporting, and data export
- Establish a framework to integrate more complex forecasts or simulations in the future

Time for a dashboard?

Case study: Flight delay dashboard



Flight Delay Report

Month

Jan

Jan Report

Average Departure Delay

In this month dep_delay exceeded the average daily threshhold of 10 a total of 10 days

Average Arrival Delay

In this month arr_delay exceeded the average daily threshhold of 10 a total of 3 days

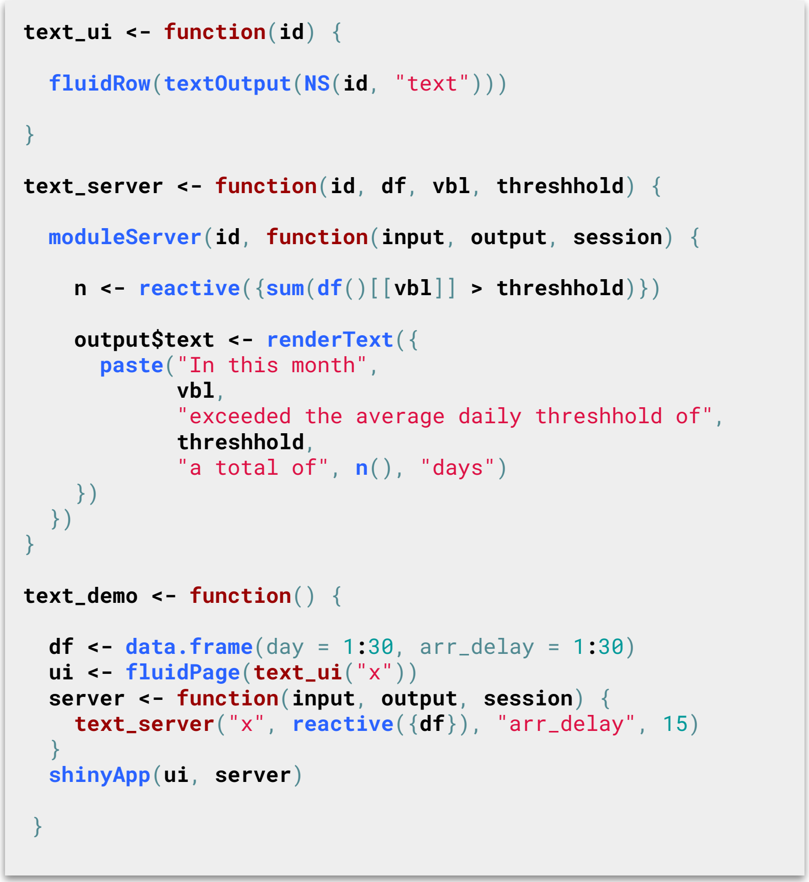
1. Decompose requirements

- Let users **pick** a month of interest to visualize

- **For each** metric of interest, users should:
  - See a time-series plot of the average daily value of the metric
  - Click a download button to download a PNG of the plot
  - Read a text summary that reports the number of days with breaches

- The metrics of interest are:
  - Average departure delay
  - Average arrival delay
  - Proportion of flights with an arrival delays >5 minutes

2. Build and test 'baby' applications

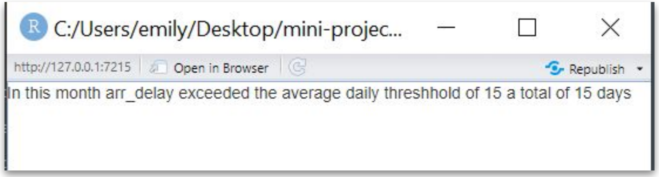


```r
text_ui <- function(id) {

  fluidRow(textOutput(NS(id, "text")))

}

text_server <- function(id, df, vbl, threshhold) {

  moduleServer(id, function(input, output, session) {

    n <- reactive({sum(df()[[vbl]] > threshhold)})

    output$text <- renderText({
      paste("In this month",
            vbl,
            "exceeded the average daily threshhold of",
            threshhold,
            "a total of", n(), "days")
    })
  })
}

text_demo <- function() {

  df <- data.frame(day = 1:30, arr_delay = 1:30)
  ui <- fluidPage(text_ui("x"))
  server <- function(input, output, session) {
    text_server("x", reactive({df}), "arr_delay", 15)
  }
  shinyApp(ui, server)

 }
```

2. Build and test ‘baby’ applications

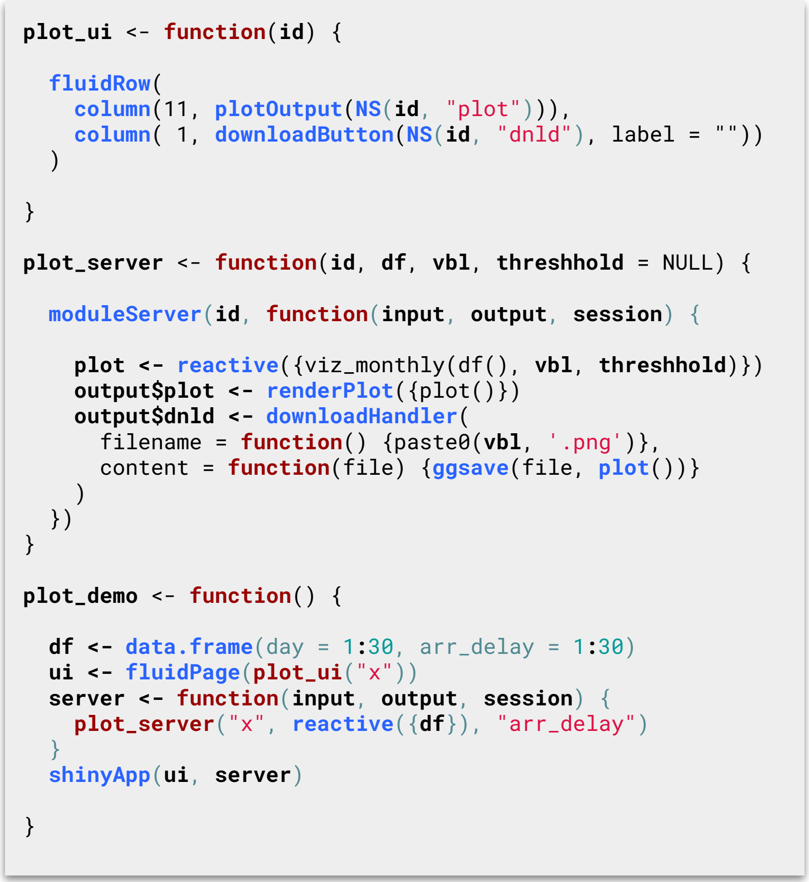
```r
text_ui <- function(id) {

  fluidRow(textOutput(NS(id, "text")))

}

text_server <- function(id, df, vbl, threshhold) {

  moduleServer(id, function(input, output, session) {

    n <- reactive({sum(df()[[vbl]] > threshhold)})

    output$text <- renderText({
      paste("In this month",
            vbl,
            "exceeded the average daily threshhold of",
            threshhold,
            "a total of", n(), "days")
    })
  })
}

text_demo <- function() {

  df <- data.frame(day = 1:30, arr_delay = 1:30)
  ui <- fluidPage(text_ui("x"))
  server <- function(input, output, session) {
    text_server("x", reactive({df}), "arr_delay", 15)
  }
  shinyApp(ui, server)

 }
```

```
> text_demo()
```



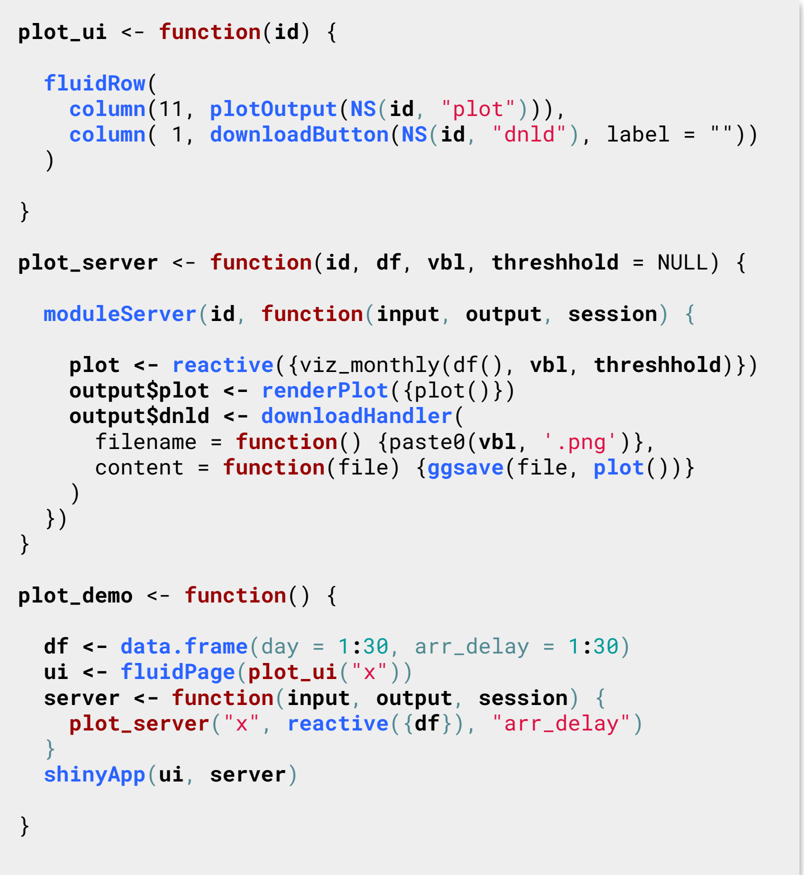
```
In this month arr_delay exceeded the average daily threshhold of 15 a total of 15 days
```

2. Build and test ‘baby’ applications



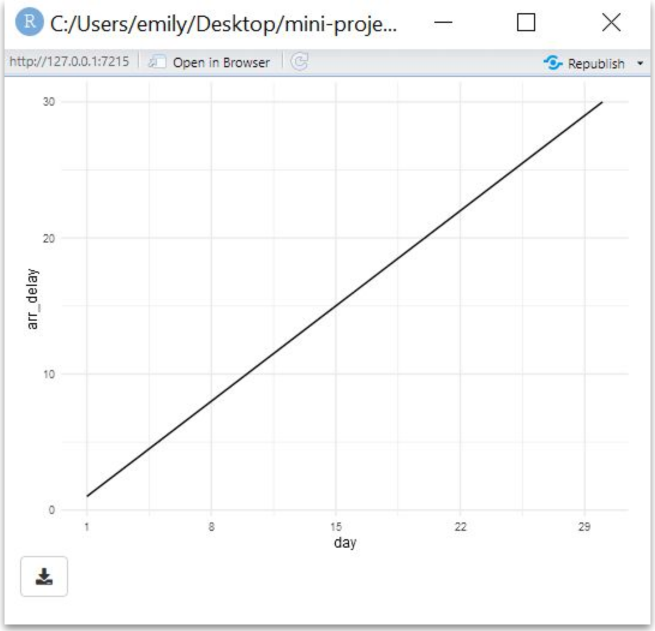
```r
plot_ui <- function(id) {

  fluidRow(
    column(11, plotOutput(NS(id, "plot"))),
    column( 1, downloadButton(NS(id, "dnld"), label = ""))
  )

}

plot_server <- function(id, df, vbl, threshhold = NULL) {

  moduleServer(id, function(input, output, session) {

    plot <- reactive({viz_monthly(df(), vbl, threshhold)})
    output$plot <- renderPlot({plot()})
    output$dnld <- downloadHandler(
      filename = function() {paste0(vbl, '.png')},
      content = function(file) {ggsave(file, plot())}
    )
  })
}

plot_demo <- function() {

  df <- data.frame(day = 1:30, arr_delay = 1:30)
  ui <- fluidPage(plot_ui("x"))
  server <- function(input, output, session) {
    plot_server("x", reactive({df}), "arr_delay")
  }
  shinyApp(ui, server)

}
```

2. Build and test ‘baby’ applications

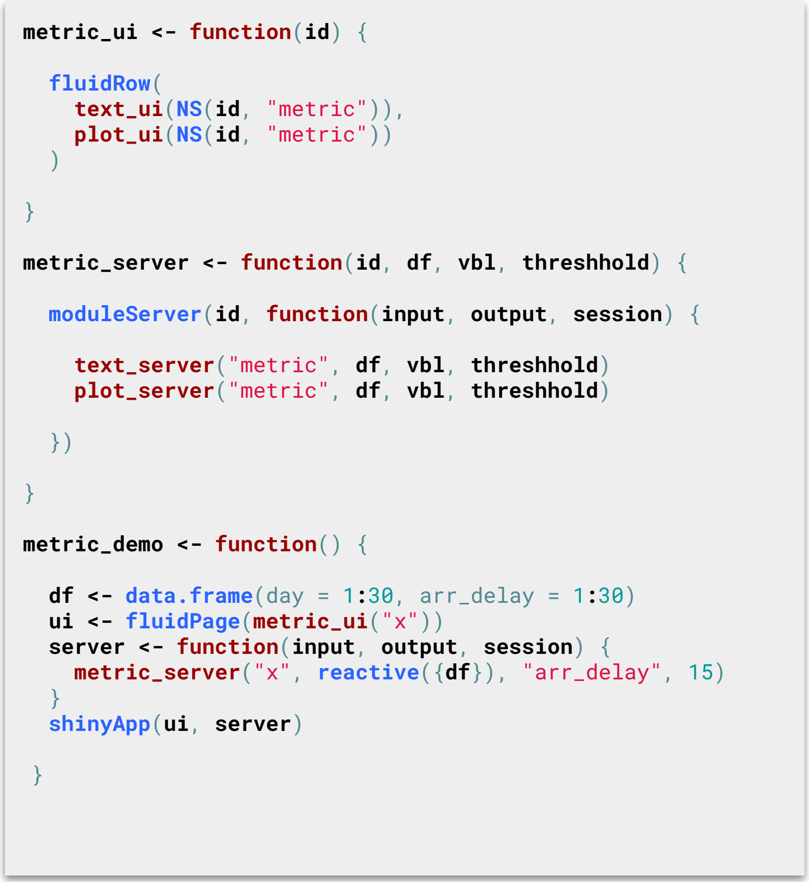


```r
plot_ui <- function(id) {

  fluidRow(
    column(11, plotOutput(NS(id, "plot"))),
    column( 1, downloadButton(NS(id, "dnld"), label = ""))
  )

}

plot_server <- function(id, df, vbl, threshhold = NULL) {

  moduleServer(id, function(input, output, session) {

    plot <- reactive({viz_monthly(df(), vbl, threshhold)})
    output$plot <- renderPlot({plot()})
    output$dnld <- downloadHandler(
      filename = function() {paste0(vbl, '.png')},
      content = function(file) {ggsave(file, plot())}
    )
  })
}

plot_demo <- function() {

  df <- data.frame(day = 1:30, arr_delay = 1:30)
  ui <- fluidPage(plot_ui("x"))
  server <- function(input, output, session) {
    plot_server("x", reactive({df}), "arr_delay")
  }
  shinyApp(ui, server)

}
```

```
> plot_demo()
```



3. Compose building blocks

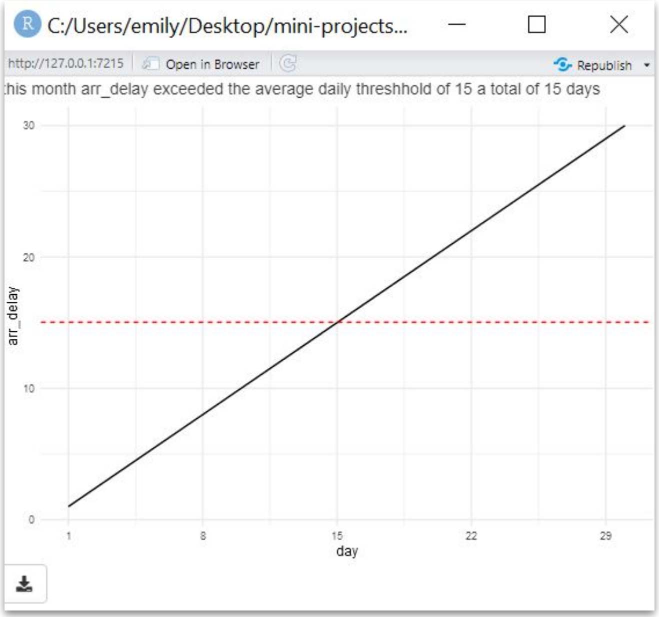


```r
metric_ui <- function(id) {

  fluidRow(
    text_ui(NS(id, "metric")),
    plot_ui(NS(id, "metric"))
  )

}

metric_server <- function(id, df, vbl, threshhold) {

  moduleServer(id, function(input, output, session) {

    text_server("metric", df, vbl, threshhold)
    plot_server("metric", df, vbl, threshhold)

  })

}

metric_demo <- function() {

  df <- data.frame(day = 1:30, arr_delay = 1:30)
  ui <- fluidPage(metric_ui("x"))
  server <- function(input, output, session) {
    metric_server("x", reactive({df}), "arr_delay", 15)
  }
  shinyApp(ui, server)

 }
```

3. Compose building blocks

```
metric_ui <- function(id) {

  fluidRow(
    text_ui(NS(id, "metric")),
    plot_ui(NS(id, "metric"))
  )

}

metric_server <- function(id, df, vbl, threshhold) {

  moduleServer(id, function(input, output, session) {

    text_server("metric", df, vbl, threshhold)
    plot_server("metric", df, vbl, threshhold)

  })

}

metric_demo <- function() {

  df <- data.frame(day = 1:30, arr_delay = 1:30)
  ui <- fluidPage(metric_ui("x"))
  server <- function(input, output, session) {
    metric_server("x", reactive({df}), "arr_delay", 15)
  }
  shinyApp(ui, server)

 }
```

```
> metric_demo()
```

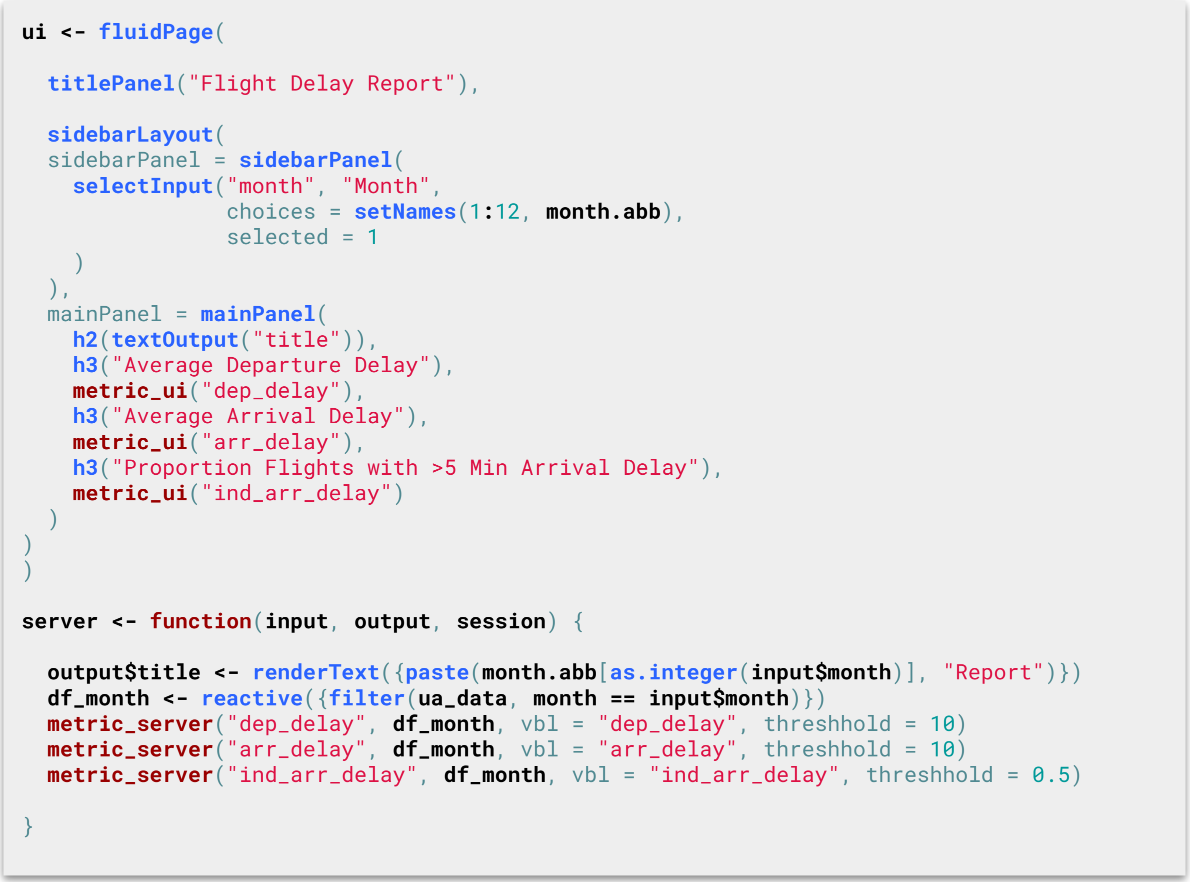


C:/Users/emily/Desktop/mini-projects...

http://127.0.0.1:7215

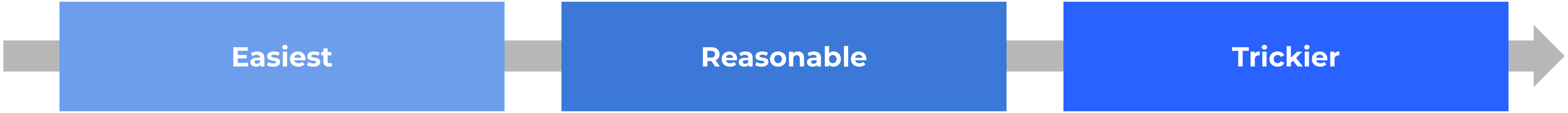
this month arr_delay exceeded the average daily threshhold of 15 a total of 15 days

4. Complete application



```
ui <- fluidPage(

  titlePanel("Flight Delay Report"),

  sidebarLayout(
  sidebarPanel = sidebarPanel(
    selectInput("month", "Month",
                choices = setNames(1:12, month.abb),
                selected = 1
    )
  ),
  mainPanel = mainPanel(
    h2(textOutput("title")),
    h3("Average Departure Delay"),
    metric_ui("dep_delay"),
    h3("Average Arrival Delay"),
    metric_ui("arr_delay"),
    h3("Proportion Flights with >5 Min Arrival Delay"),
    metric_ui("ind_arr_delay")
  )
)
)

server <- function(input, output, session) {

  output$title <- renderText({paste(month.abb[as.integer(input$month)], "Report")})
  df_month <- reactive({filter(ua_data, month == input$month)})
  metric_server("dep_delay", df_month, vbl = "dep_delay", threshhold = 10)
  metric_server("arr_delay", df_month, vbl = "arr_delay", threshhold = 10)
  metric_server("ind_arr_delay", df_month, vbl = "ind_arr_delay", threshhold = 0.5)

}
```

Not all module patterns are equally beginner-friendly



Easiest

Reasonable

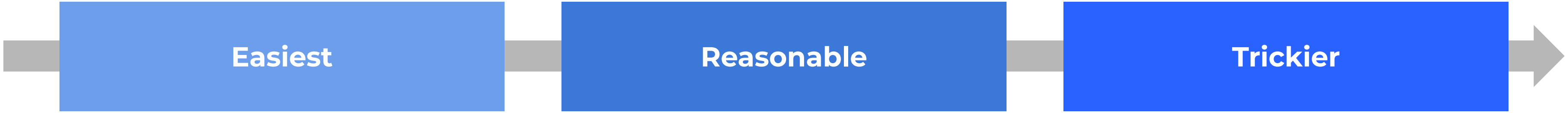
Trickier

Given reactive input
Generate output

Given reactive input + config
Generate output

Given reactive input &| config
Pass back reactives

Not all module patterns are equally beginner-friendly



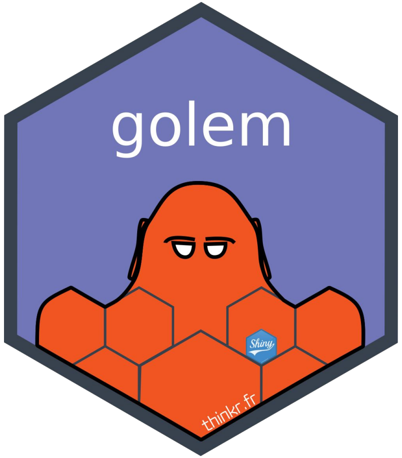
Given reactive input
Generate output

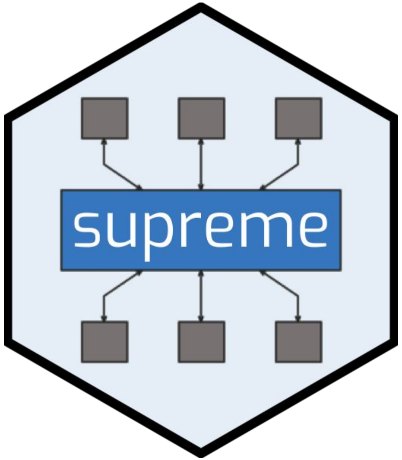
Given reactive input + config
Generate output

Given reactive input &| config
Pass back reactives

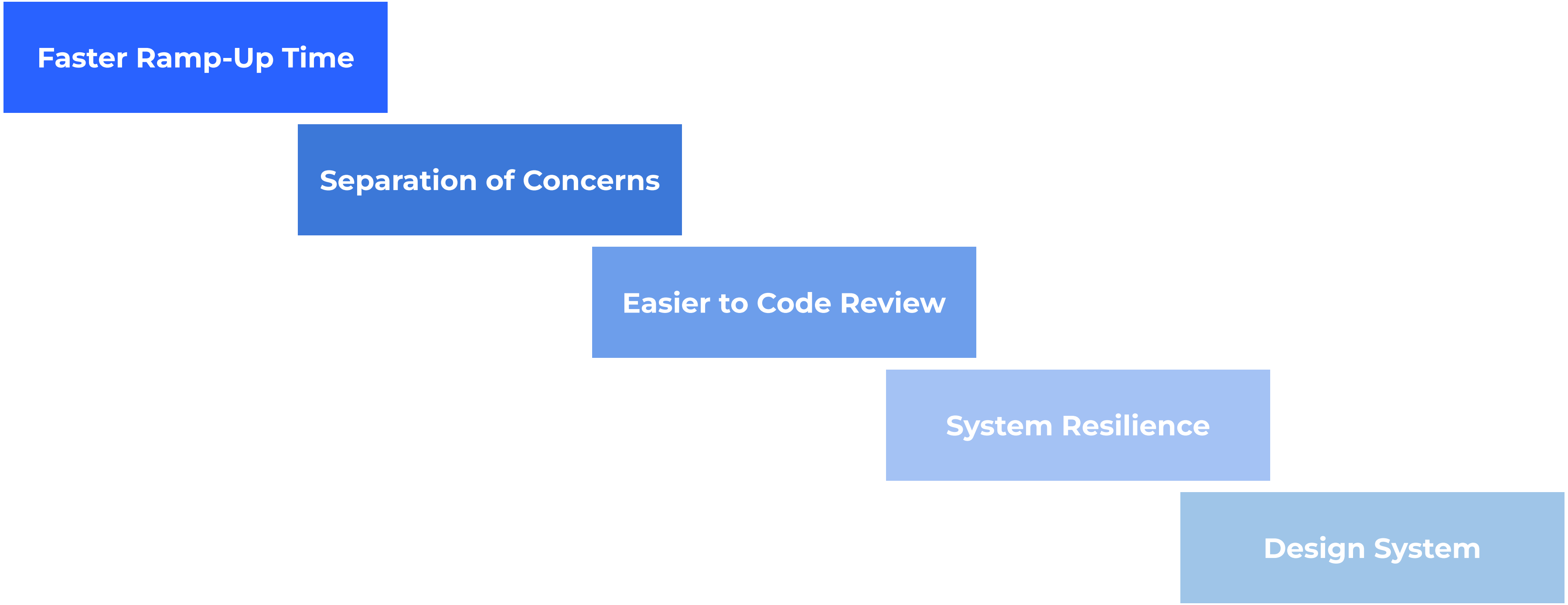
⭐
Recall Marcin Dubel's session on best practices for this!
⭐

Modules help set the stage for more advanced workflows





Modular approaches can be even more crucial to onboarding and collaborating with colleagues in enterprise settings



Faster Ramp-Up Time

Separation of Concerns

Easier to Code Review

System Resilience

Design System

New developers can be cruising in no time with modules!

Photo Credit: John McArthur on Unsplash

Questions?

↓ Read more ↓

Code on GitHub

Blog Post

Mastering Shiny book

Documentation