Welcome to instats

The Session Will Begin Shortly

START

Spatial Data Analysis and Visualization in R

Session 22: Integrating tmap with the R Package Shiny for Dashboards

instats

Overview

- Shiny basics
- Static vs interactive `tmap`
- Live map mode
- Dynamic control with `tm_basemap()`
- Combining `tmap` and `Shiny` UI widgets

What is Shiny?

Shiny is an R package that lets you build interactive web apps using only R.

- UI and server structure
- Reactive programming model
- Useful for data dashboards, maps, and more

Shiny App Skeleton

```r
library(shiny)

ui <- fluidPage(
    h2("My Shiny App"),
    textInput("txt", "Type something"),
    textOutput("output")
)

server <- function(input, output) {
    output$output <- renderText({
        paste("You typed:", input$txt)
    })
}

shinyApp(ui, server)
```

Example: Simple Interactive Map

```r
library(tmap)
library(spData)

tmap_mode("view")  ## enable interactive mode

tm_shape(World) +
    tm_borders() +
    tm_fill("continent")
```

Example: map with `renderTmap()` in Shiny

```r
library(shiny)
library(tmap)
library(spData)

tmap_mode("view")

ui <- fluidPage(
    tmapOutput("map")
)

server <- function(input, output) {
    output$map <- renderTmap({
        tm_shape(World) +
            tm_borders() +
            tm_fill("pop_est_dens")
    })
}

shinyApp(ui, server)
```

Example: map with input widgets

Use `selectInput()` to dynamically control the fill variable.

```r
tmap_mode("view")

ui <- fluidPage(
    selectInput("var", "Variable", choices = c("pop_est_dens", "gdp_cap_est")),
    tmapOutput("map")
)

server <- function(input, output) {
    output$map <- renderTmap({
        tm_shape(World) +
            tm_fill(input$var)
    })
}

shinyApp(ui, server)
```

Update the map with `tmapProxy()`

- In the previous example, the map will *rerender* after selecting another variable
- Better is to *update* the map, which will retain the current view
- Only useful in *view* mode

Example with `tmapProxy()`

```r
world_vars <- setdiff(names(World), c("iso_a3", "name", "sovereignt", "geometry"))
tmap_mode("view")
shinyApp(
    ui = fluidPage(
        tmapOutput("map", height = "600px"), selectInput("var", "Variable", world_vars)),
    server <- function(input, output, session) {
        output$map <- renderTmap({
            tm_shape(World, id = "iso_a3") + tm_polygons(fill = world_vars[1], zindex = 401)
        })
        observe({
            var <- input$var
            tmapProxy("map", session, {
                tm_remove_layer(401) +
                    tm_shape(World, id = "iso_a3") +
                    tm_polygons(fill = var, zindex = 401)
            })
        })
    }, options = list(launch.browser=TRUE)
)
```

Recap

- `tmap` integrates easily with Shiny via `renderTmap()` and `tmapOutput()`
- Use reactive inputs like `selectInput()` to control map content
- Use `tmapProxy()` to update the map (view mode only)
- Set the tmap mode before running the app (in the global script)

STOP