Don’t Repeat Yourself: Templatize your R Shiny Apps with Modules

Cascadia R Conference

June 22, 2024

Erica Bishop

Background

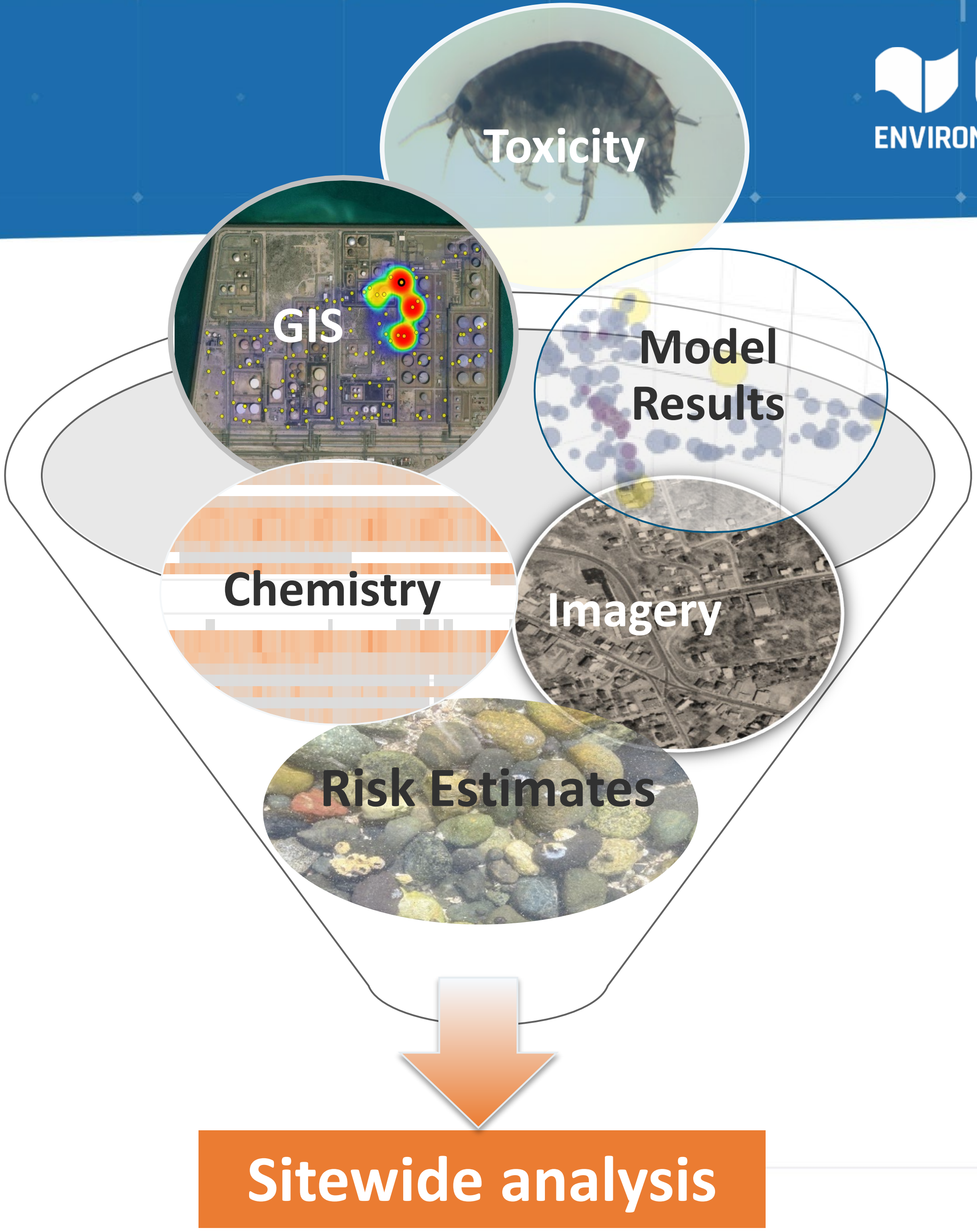
› Complex environmental data

› Lean data science team

› Quick-turnaround timelines

How we use R Shiny

- Interactivity for complex data visualization
- Internal collaboration with non-coders
- Communication with clients

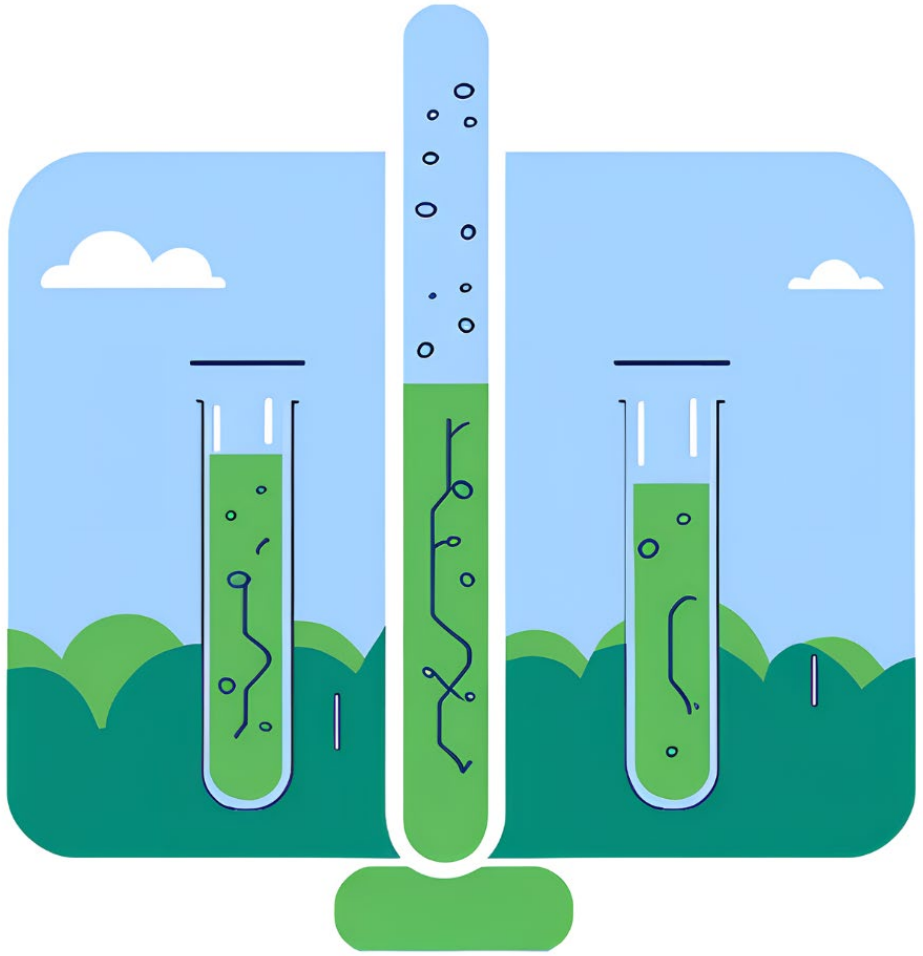


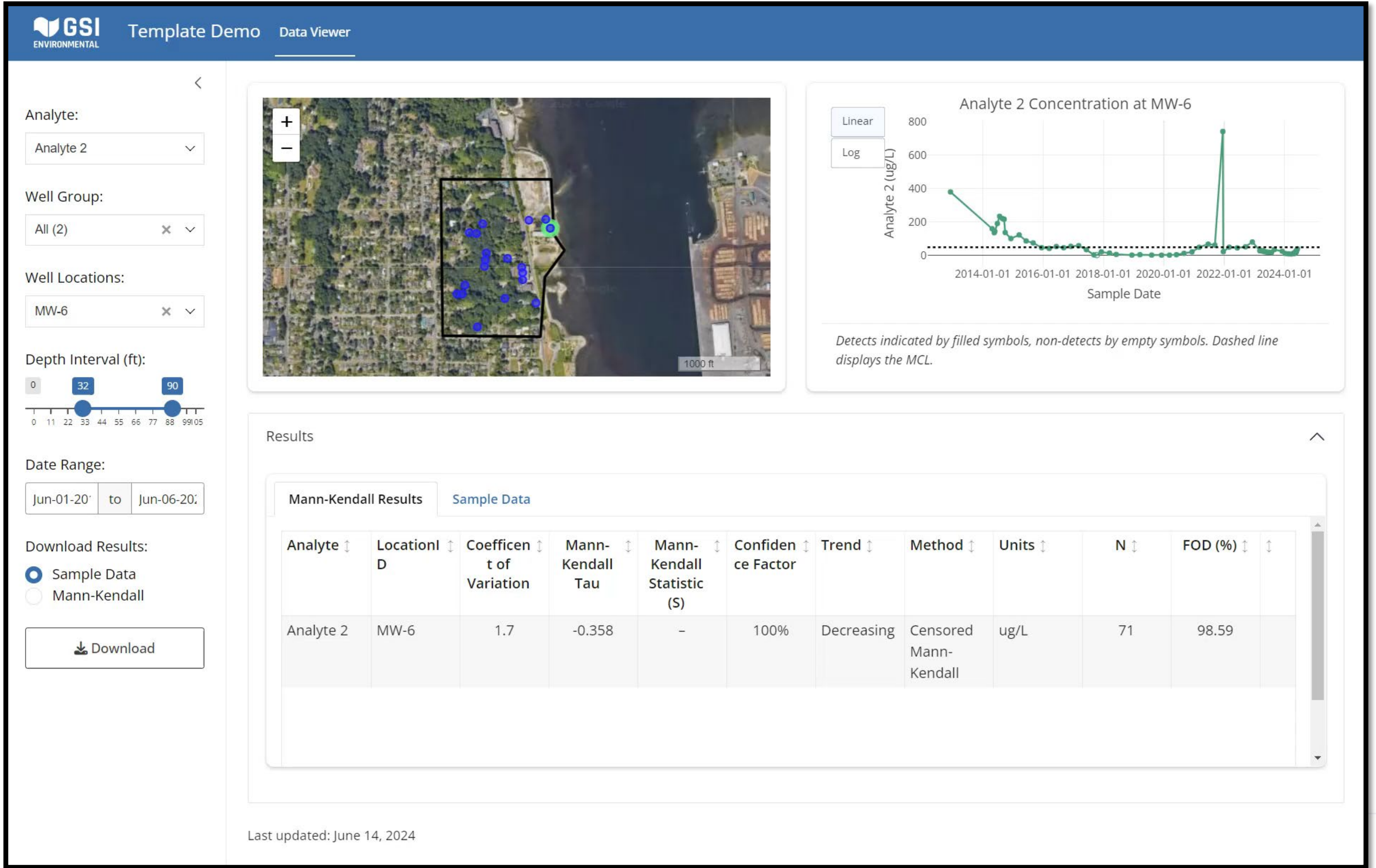
Why build a template?

- Balance customization needs with out-of-the-box tools
- Maintainable as a small team
- Easy for other R-coders to use without extra training
- Minimize overhead cost and time

Our use case

- Environmental analyses that require:
  - Filtering by chemical, location, date, depth, etc.
  - Interactive site maps
  - Timeseries plotting
  - Trend analysis
  - Statistical summary tables





Analyte: Analyte 2

Well Group: All (2)

Well Locations: MW-6

Depth Interval (ft): 32 – 90

Date Range: Jun-01-20 to Jun-06-20

Download Results:
[✓ Sample Data]
Mann-Kendall

Detects indicated by filled symbols, non-detects by empty symbols. Dashed line displays the MCL.

Mann-Kendall Results | Sample Data

Analyte	LocationID	Coefficent of Variation	Mann-Kendall Tau	Mann-Kendall Statistic (S)	Confidence Factor	Trend	Method	Units	N	FOD (%)
Analyte 2	MW-6	1.7	-0.358	–	100%	Decreasing	Censored Mann-Kendall	ug/L	71	98.59

Last updated: June 14, 2024

Basic ingredients for a Shiny template

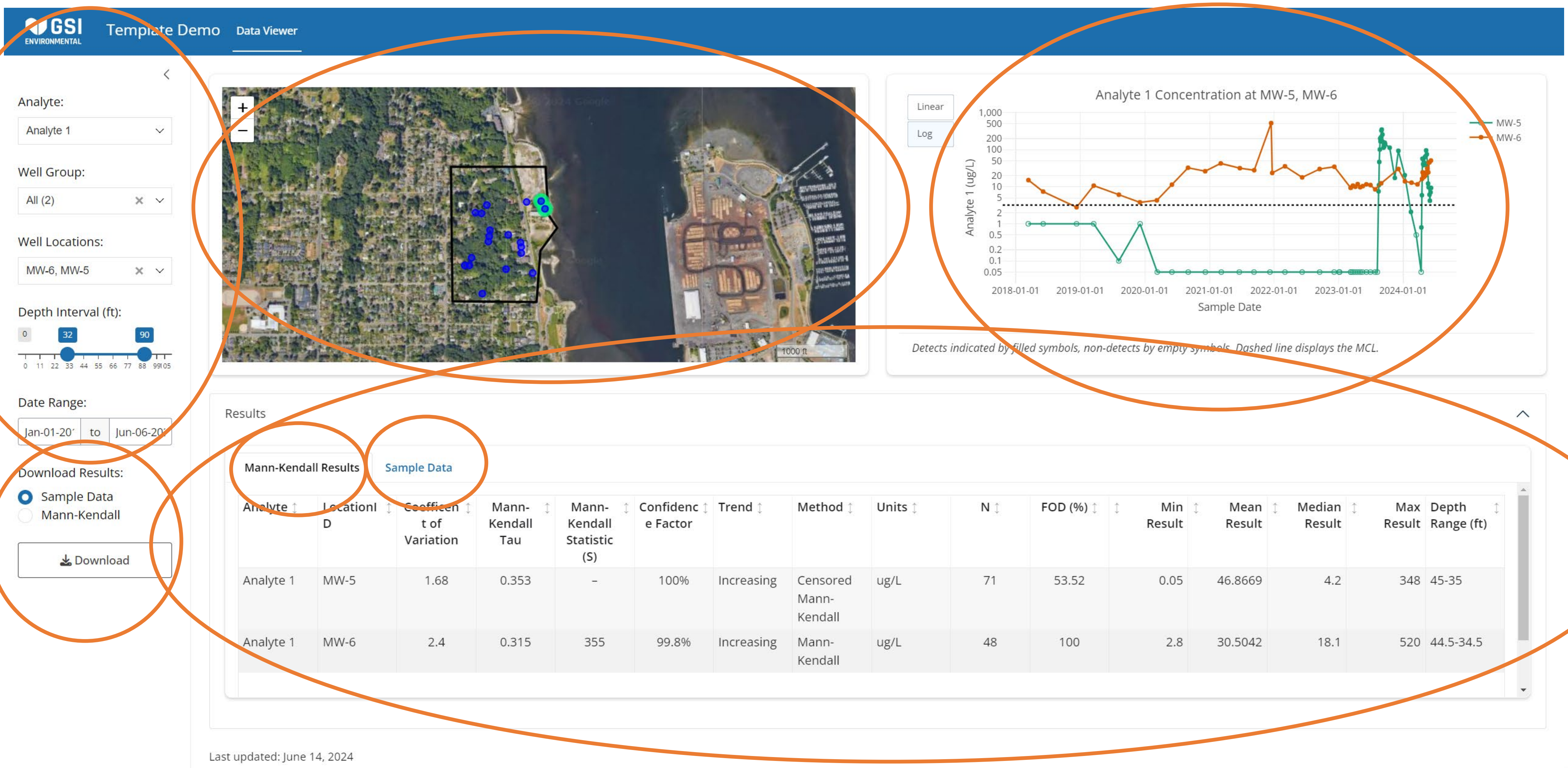
GSI ENVIRONMENTAL

MODULARITY

APP FRAMEWORK

A CENTRAL CODE BASE

Modularity



Analyte: Analyte 1

Well Group: All (2)

Well Locations: MW-6, MW-5

Depth Interval (ft): 0 32 90

Date Range: Jan-01-20 to Jun-06-20

Download Results:
[✓ Sample Data]
Mann-Kendall

Download

Detects indicated by filled symbols, non-detects by empty symbols. Dashed line displays the MCL.

Results

Mann-Kendall Results | Sample Data

Analyte	LocationID	Coefficient of Variation	Mann-Kendall Tau	Mann-Kendall Statistic (S)	Confidence Factor	Trend	Method	Units	N	FOD (%)	Min Result	Mean Result	Median Result	Max Result	Depth Range (ft)
Analyte 1	MW-5	1.68	0.353	–	100%	Increasing	Censored Mann-Kendall	ug/L	71	53.52	0.05	46.8669	4.2	348	45-35
Analyte 1	MW-6	2.4	0.315	355	99.8%	Increasing	Mann-Kendall	ug/L	48	100	2.8	30.5042	18.1	520	44.5-34.5

Last updated: June 14, 2024

Example: mod_Plot

Importing packages and modules the rhino way

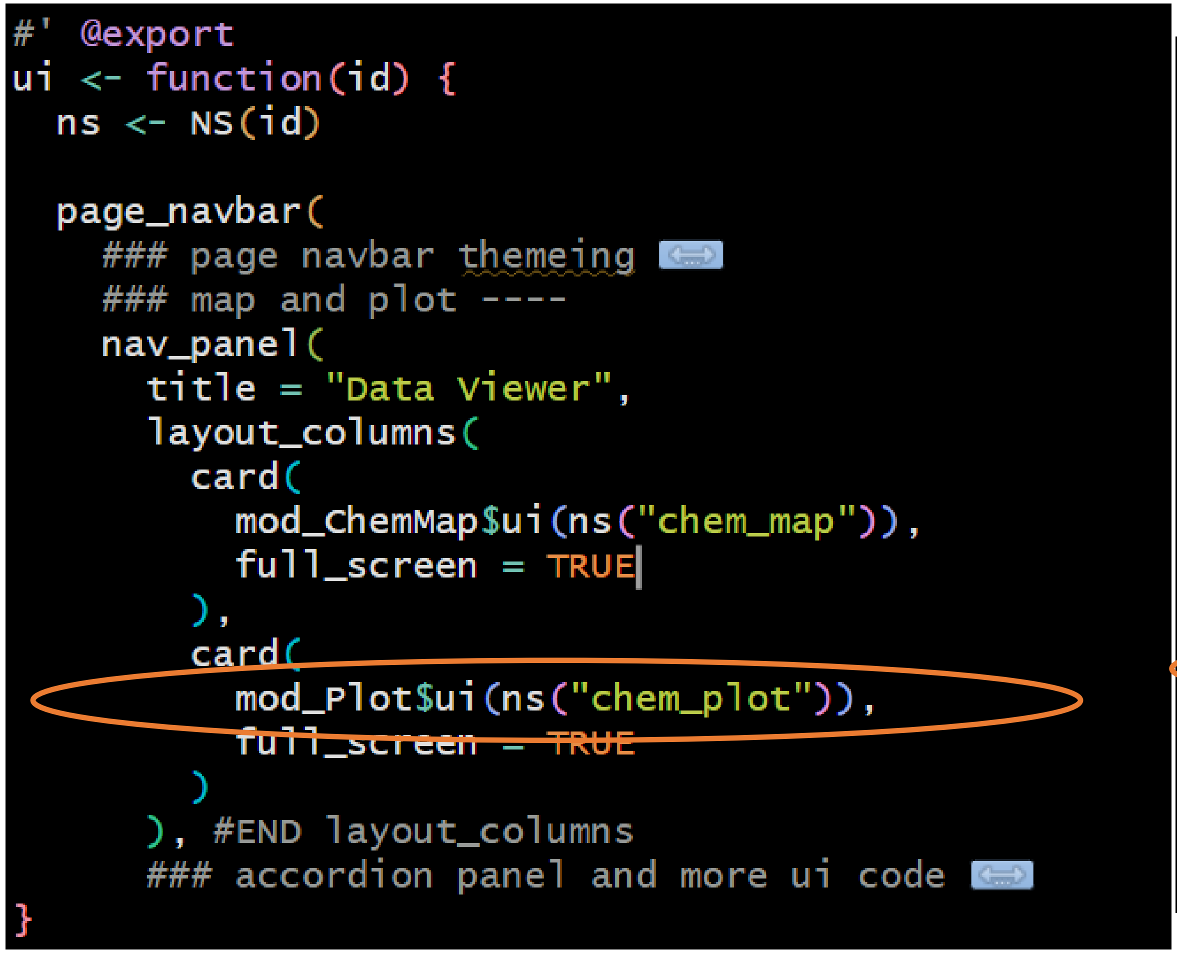
Called with `mod_Plot$ui`

Called inside `moduleServer()` with `mod_Plot$server`

```r
# app/view/mod_Plot
box::use(
  shiny[...],
  bslib[...],
  dplyr[arrange, mutate],
  plotly
)

box::use(
  app/logic/fun_plotly
)

#' @export
ui <- function(id) {
  ns <- NS(id)
  tagList(
    plotly$plotlyOutput(ns("chem_plot")),
    card_footer(
      tags$i("Card footer text.")
    )
  )
}

#' @export

server <- function(id, rv_df_anl){
  moduleServer(id, function(input, output, session){

    # Render Plotly -------------
    output$chem_plot <- plotly$renderPlotly({

      p <- plotly$plot_ly()
    }) #END renderPlotly

  }) #END moduleServer
}
```

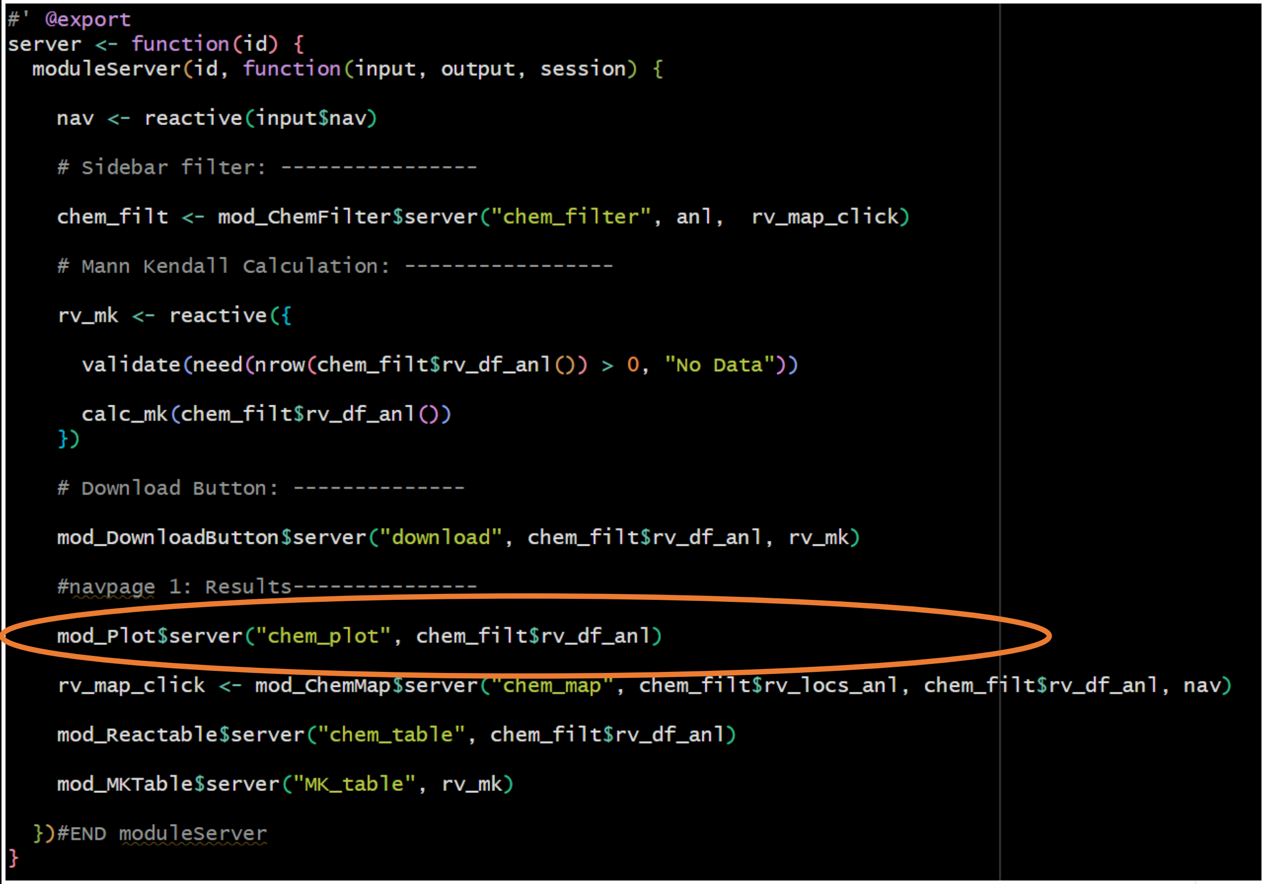
Modularity

ui function



```
#' @export
ui <- function(id) {
  ns <- NS(id)

  page_navbar(
    ### page navbar themeing
    ### map and plot ----
    nav_panel(
      title = "Data Viewer",
      layout_columns(
        card(
          mod_ChemMap$ui(ns("chem_map")),
          full_screen = TRUE
        ),
        card(
          mod_Plot$ui(ns("chem_plot")),
          full_screen = TRUE
        )
      ), #END layout_columns
      ### accordion panel and more ui code
}
```

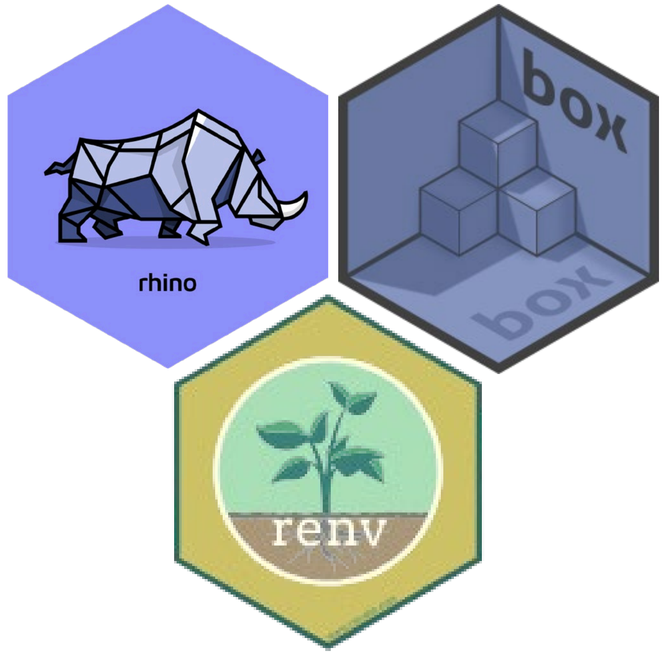
moduleServer()



```
#' @export
server <- function(id) {
  moduleServer(id, function(input, output, session) {

    nav <- reactive(input$nav)

    # Sidebar filter: ----------------

    chem_filt <- mod_ChemFilter$server("chem_filter", anl,  rv_map_click)

    # Mann Kendall Calculation: -----------------

    rv_mk <- reactive({

      validate(need(nrow(chem_filt$rv_df_anl()) > 0, "No Data"))

      calc_mk(chem_filt$rv_df_anl())
    })

    # Download Button: --------------

    mod_DownloadButton$server("download", chem_filt$rv_df_anl, rv_mk)

    #navpage 1: Results---------------

    mod_Plot$server("chem_plot", chem_filt$rv_df_anl)

    rv_map_click <- mod_ChemMap$server("chem_map", chem_filt$rv_locs_anl, chem_filt$rv_df_anl, nav)

    mod_Reactable$server("chem_table", chem_filt$rv_df_anl)

    mod_MKTable$server("MK_table", rv_mk)

  })#END moduleServer
}
```

Framework

- Rhino is our framework of choice because it handles:
  - Apps that scale up
  - Modular structure
  - Consistency across every app
  - Version control (with renv)



Storing the template with GitHub

Accessible

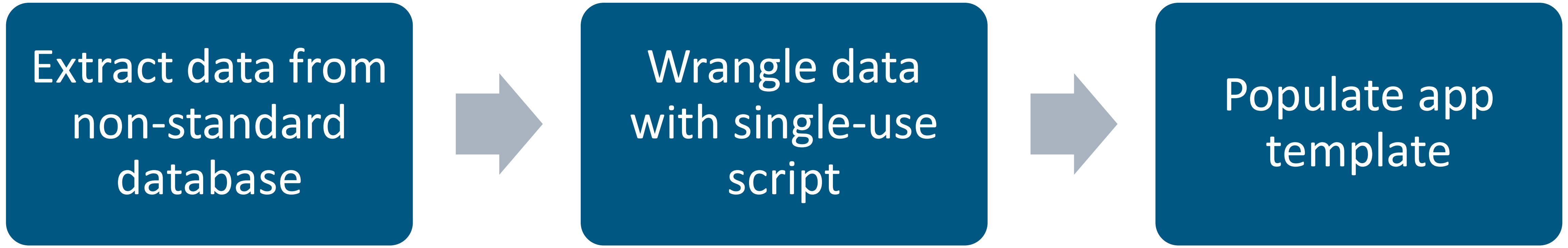
Anyone on the team can contribute new modules or changes to the template

Centralized

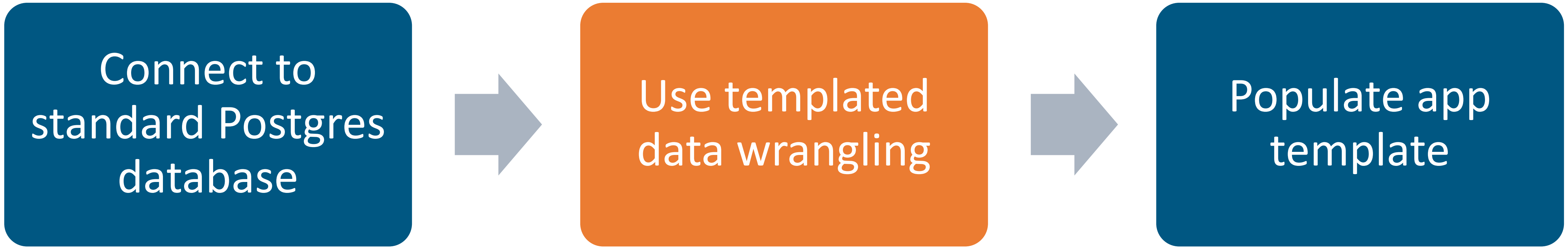
codebase for shiny and data-viz related tactics

Integrating with data management

Time-intensive!


Extract data from non-standard database
Wrangle data with single-use script
Populate app template

Integrating with data management


Connect to standard Postgres database
Use templated data wrangling
Populate app template

Consistent coding practices matter

GSI ENVIRONMENTAL

- Reactive variable naming conventions
- Clean code structure
- File naming to follow the rhino structure

The bottom line:

A template is a lightweight workflow for saving time and improving collaboration on R Shiny app development

THANK YOU

GSI ENVIRONMENTAL

Science · Strategy · Solutions

Erica Bishop

enbishop@gsi-net.com