Data Storytelling Dashboard for Exploring Auckland Air Quality

Stephen Su

STATS 781

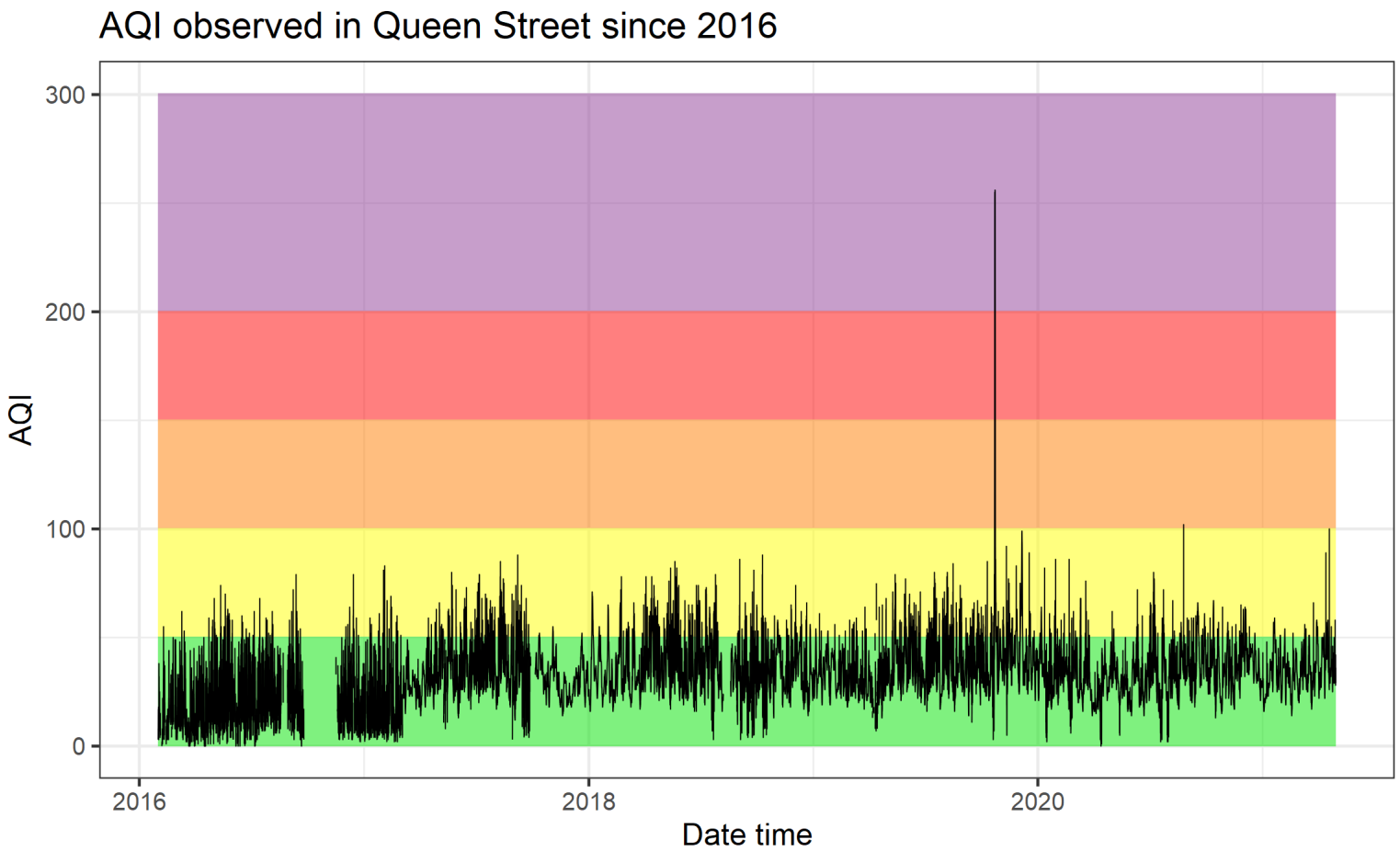
Auckland air quality

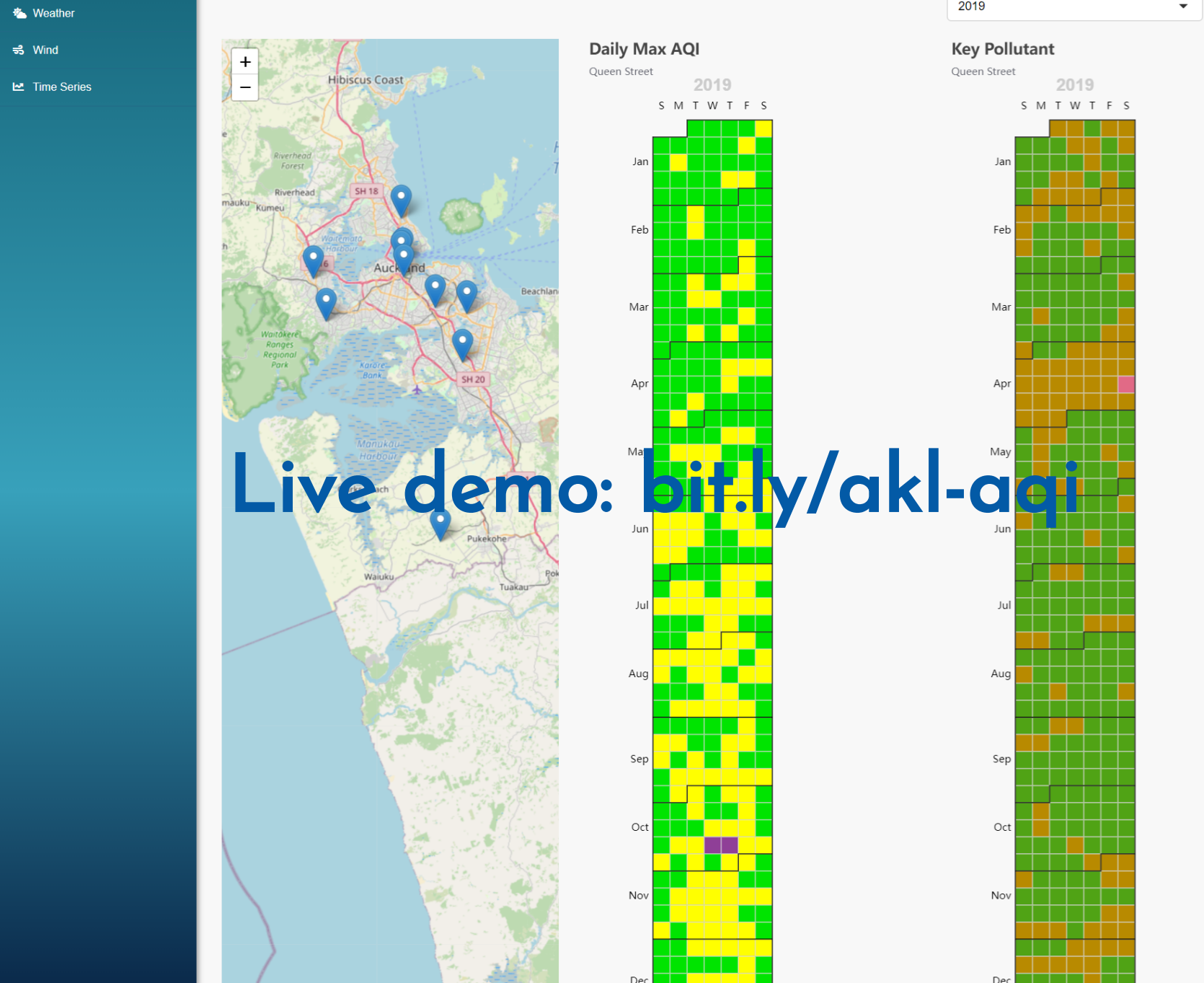
- New Zealand is well-known for the clean air.
- Maintaining the reputation needs continuous monitoring.
  - Auckland Council's air quality management plan
- Main metric: air quality index (AQI)

AQI Level of Concern	Value of Index	Colour (Hexadecimal Code)
Good	0 to 50	Green (#00E400)
Moderate	51 to 100	Yellow (#FFFF00)
Unhealthy for sensitive groups	101 to 150	Orange (#FF7E00)
Unhealthy	151 to 200	Red (#FF0000)
Very unhealthy	201 to 300	Purple (#8F3F97)
Hazardous	301 and higher	Maroon (#7E0023)

- Data source: Auckland Regional Council

A static visualisation

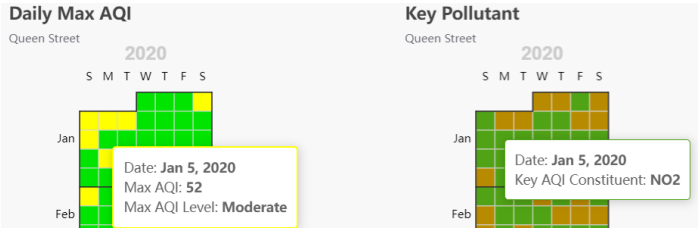


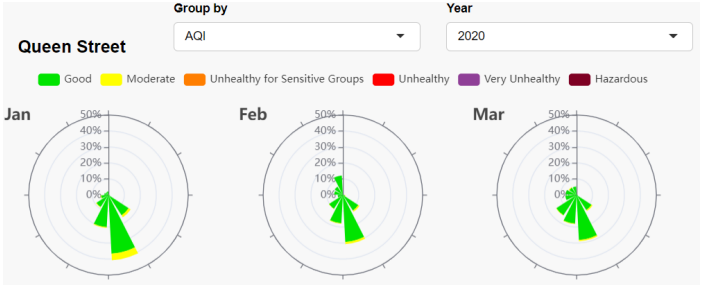


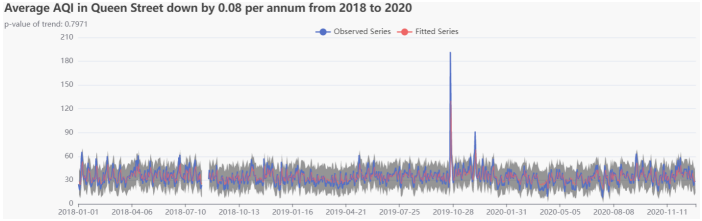
Live demo: bit.ly/akl-aqi

Interactive graphics implemented

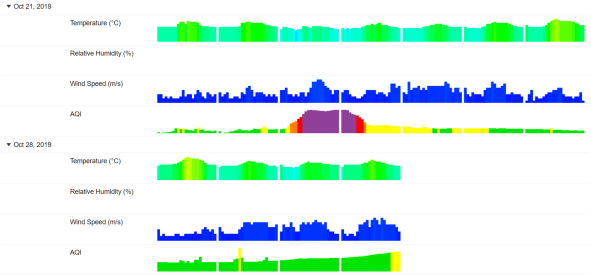
echarts4r



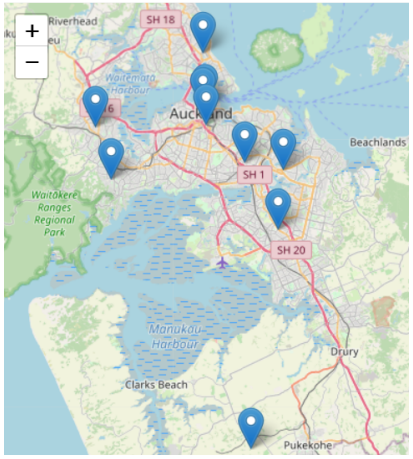




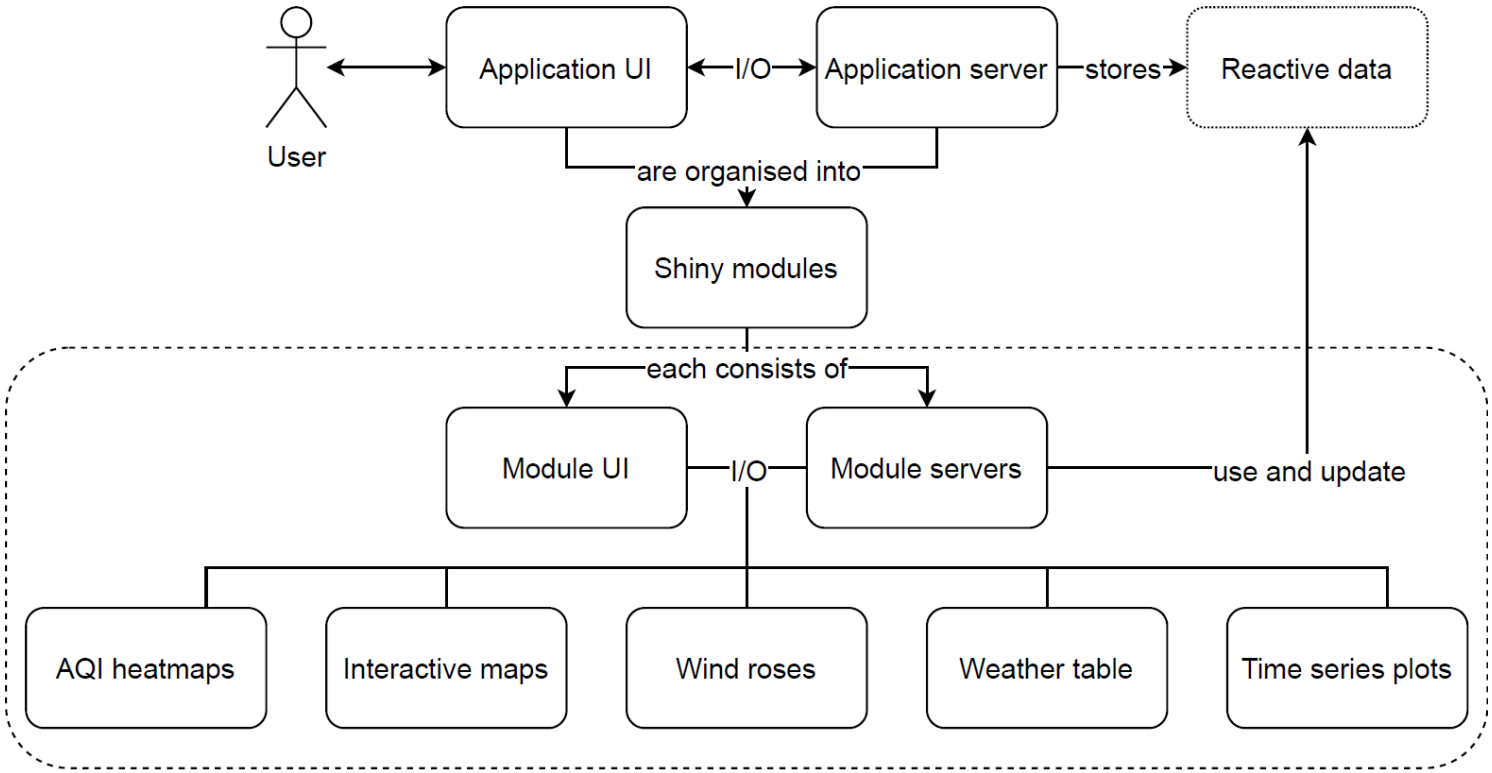
reactable



leaflet



A modularised shiny application



Shiny modules

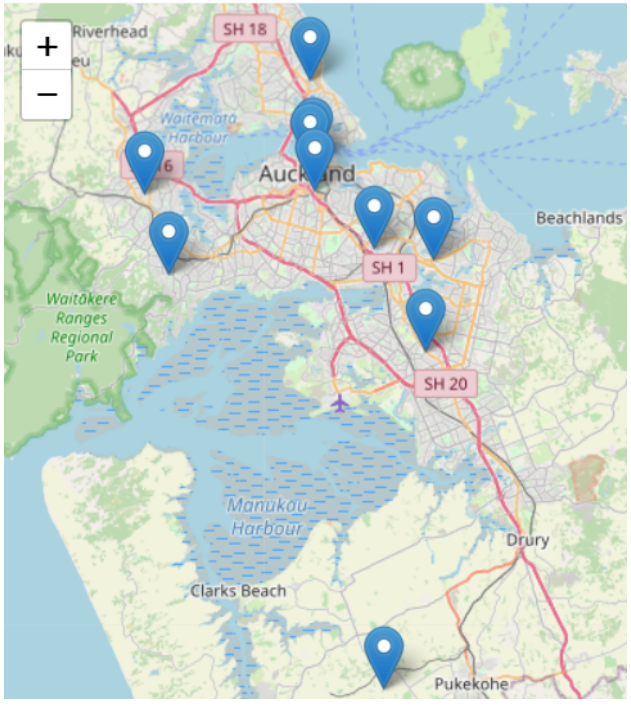
If the application is developed as a whole, the codebase would become chunky and unmanageable.

```
e <- e_charts(data, ...) # Data for which chart?
e_chart_1 <- echarts(data_for_chart_1, ...)
e_chart_2 <- echarts(data_for_chart_2, ...)
```

Modularisation comes to rescue

- Each interactive graphic is encapsulated as a module.
- The dashboard consists of a collection of interactive graphics.
- Modularisation facilitates easy maintenance and extensibility

Shiny module example: the map




```r
map_aqi_ui <- function(id) {
  # Unique namespace
  ns <- NS(id)
  # Call for graphics output
  leafletOutput(ns("map_aqi"))
}

map_aqi_mod <-
  function(id, state) {
    module <-
      function(...) {
        ## Render graphics output
      }
    # Run with unique namespace
    moduleServer(id, module)
  }
```

A shiny module

- Consists of its own UI and server
- Represents an encapsulated R environment
  - Has dedicated and isolated namespace
  - Does not interfere with other modules

```r
map_aqi_ui <- function(id) {
  # Unique namespace
  ns <- NS(id)
  # Call for graphics output
  leafletOutput(ns("map_aqi"))
}

map_aqi_mod <-
  function(id, state) {
    module <-
      function(...) {
        ## Render graphics output
      }
    # Run with unique namespace
    moduleServer(id, module)
  }
```

Assembling modules

Application server

```
app_server <- function(input, output, session) {
  ## Other shiny module servers
  map_aqi_mod("map_aqi", app_state)
  ## Other server codes
}
```

Application UI

```
app_ui <- dashboardPage(
  header, sidebar,
  body = dashboardBody(
    tabItem(tabName,
      fluidRow(column(map_aqi_ui("map_aqi"))),
      ## Other modular UI
    ),
    ## Other tabs
  )
)
```

If the modules are isolated, how do they communicate?

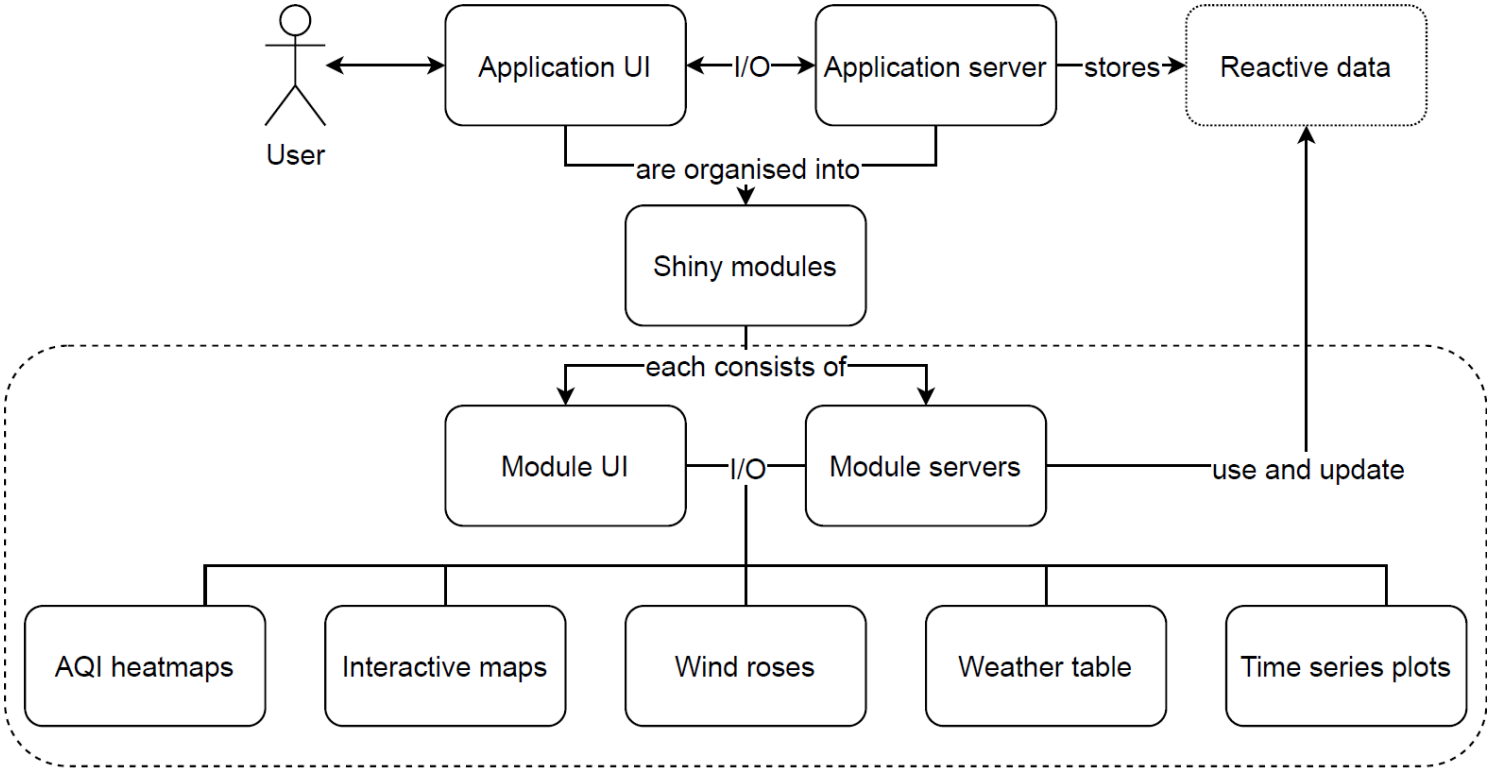
Event-driven module communication

- A shiny reactive `app_state` stores the snapshot of the current session.

```
app_state <- reactiveValues(data, user_clicks, cached_graphics, ...)
```

- The shiny application "reacts" to events;
    - e.g., when a user clicks on a station on the interactive map:

```
map_aqi_mod <- function(id, state) {
  module <- function(input, output, session) {
    ## Shiny output
    observeEvent(input[["map_aqi_marker_click"]], {
      state[["map_onclick"]] <- input[["map_aqi_marker_click"]][["id"]]
    })
  }
  moduleServer(id, module)
}
```


User
Application UI
I/O
Application server
stores
Reactive data
are organised into
Shiny modules
each consists of
Module UI
I/O
Module servers
use and update
AQI heatmaps
Interactive maps
Wind roses
Weather table
Time series plots

Shiny reactive 👆

```
app_state <- reactiveValues(data, user_clicks, cached_graphics, ...)
```

- Has only one collection of values at any time;
- Is evaluated only when its value is updated;
- Is cached;
- Can be updated by "reacting" to an event.

BTS: How can we keep the app performant with > 1M observations of data?

Data caching

- Loads data on demand
- Uses reactives
- Detects user clicks as events
- Checks if data for the clicked-location is loaded
- Load data if not already loaded

```
append_data <-
  function(data, loc) {
    ## Load new data for loc,
    ## then return new data set
    ## by binding new data
    ## to the old data
  }

app_server <- function(...) {
  ## Shiny module servers
  ## Other server codes
  observeEvent(map_click, {
    if (!data_is_loaded) {
      app_state[["data"]] <-
        append_data(...)
    }
  })
}
```

Graphics caching

- Saves rendering time
- Temporarily saving rendered graphics for the session in the memory
- Uses shiny function `bindCache()`

```
aqi_heatmap_mod <- function(id, state = app_state) {
  module <- function(input, output, session) {
    ## Reactivity and event handling
    ## Data processing
    output[["aqi_heatmap"]] <- renderEcharts4r(expr) %>%
      bindCache(...)
  }
  moduleServer(id, module)
}
```

Project contribution

- This project delivers an insightful storytelling dashboard.
- It is accessible to the public via bit.ly/akl-aqi.
- Approaches exploration of Auckland air quality in multiple aspects.
- Various interactive graphics are integrated into one web application.
- The graphics are modularised but linked with user interaction.

Open and reproducible research

- Dissertation: https://github.com/szmsu2011/hons-dissertation
- Application source code: https://github.com/szmsu2011/akl-air-quality

Explore on your own

bit.ly/akl-aqi