Exercise V – Advanced Shiny

Part 2 – Advanced Shiny UI

CEE412 / CET522

TRANSPORTATION DATA MANAGEMENT AND VISUALIZATION

WINTER 2020

Outline

- DT – DataTable
- Leaflet – Map data visualization
- dygrpahs for R – Time series data visualization
- Other visualization tools
  - networkD3
  - metricsgraphics
- Dashboard:
  - flexdashboard
- Other Notes

Demo Dataset

- In this exercise, we use a Car2Go trip dataset to demonstrate some data visualization tools.
- Dataset: Car2Go trip data in Seattle area, collected in 2016
  - Server: 128.95.29.72, Database: E5_Car2GoData
    - Columns:
      - Id: vehicle ID (License Plate ID #)
      - otime, dtime: starting and ending time of a trip
      - olon, olat: coordinates of the origin of a trip
      - dlon, dlat: coordinates of the destination of a trip
      - distance: distance between origin and destination (unit: mile)
      - ofuel, dfuel: vehicle's fuel status before and after a trip
      - oaddress, daddress: address of the origin and destination of a trip

DT

- The R package **DT** provides an R interface to the JavaScript library **Data Tables**.
  - R data objects (matrices or data frames) can be displayed as tables on HTML pages, and **DataTables** provides filtering, pagination, sorting, and many other features in the tables.
  - Website and demo: https://rstudio.github.io/DT/

DT

- Visualize Car2Go data in a DT table:
  - Install DT before Library DT
  - Build database connection using DBI
  - Query Top 80 rows from the data table.
  - UI:

```
ui <- fluidPage(
        DT::dataTableOutput("table1")
)
```

  - Server function:

```
output$table1 <- DT::renderDataTable(DT::datatable({
    data1 <- queryData
}))
```

```
library(shiny)
library(DBI)
library(odbc)
# Install.packages('DT')
library(DT)

conn <- DBI::dbConnect(odbc::odbc(),
                       Driver = "SQLServer",
                       Server = "128.95.29.72",
                       Database = "CEE412_CET522_W20",
                       UID      = "CEE412CET522",
                       PWD      = "Winter2020",
                       Port     = 1433)

query <- "SELECT TOP 80 * FROM [E5_Car2GoData]"
queryData <- dbGetQuery(conn, query)
```

DT

Sort values

Search values

Show 10 entries

Search:

	id	otime	dtime	olon	olat	dlon	dlat	distance	ofuel	dfuel	fuel_consumption	oaddress	daddress
1	AKN2620	2016-05-06 09:18:00.0000000	2016-05-06 09:33:30.0000000	-122.38475	47.70117	-122.38721	47.67012	2.14821	100	100	0	NW 99th St 2160, 98117 Seattle	NW 57th St 2283, 98107 Seattle
2	AKN2620	2016-05-06 10:29:30.0000000	2016-05-06 10:57:30.0000000	-122.38721	47.67012	-122.3224	47.60552	5.39222	100	100	0	NW 57th St 2283, 98107 Seattle	Jefferson St 1050, 98104 Seattle
3	AKN2620	2016-05-06 11:49:30.0000000	2016-05-06 12:15:00.0000000	-122.3224	47.60552	-122.32253	47.60457	0.06591	100	100	0	Jefferson St 1050, 98104 Seattle	Terrace St 958, 98104 Seattle
4	AKN2620	2016-05-06 14:12:00.0000000	2016-05-06 14:27:30.0000000	-122.32253	47.60457	-122.33173	47.60259	0.4511	100	100	0	Terrace St 958, 98104 Seattle	James St 271, 98104 Seattle
5	AKN2620	2016-05-06 15:10:30.0000000	2016-05-06 15:34:00.0000000	-122.33173	47.60259	-122.31317	47.55121	3.65411	100	100	0	James St 271, 98104 Seattle	S Orcas St 1527, 98108 Seattle
6	AKN2620	2016-05-06 16:19:30.0000000	2016-05-06 17:56:00.0000000	-122.31317	47.55121	-122.31239	47.55137	0.03812	100	100	0	S Orcas St 1527, 98108 Seattle	16th Ave S 5783, 98108 Seattle
7	AKN2627	2016-05-06 07:37:30.0000000	2016-05-06 08:23:00.0000000	-122.3284	47.68643	-122.34082	47.61089	5.25088	75	69	-6	1st Ave NE 7775, 98115 Seattle	Stewart St 158, 98101 Seattle
8	AKN2627	2016-05-06 08:54:30.0000000	2016-05-06 09:22:30.0000000	-122.34082	47.61089	-122.35133	47.61527	0.57674	69	69	0	Stewart St 158, 98101 Seattle	Cedar St 60, 98121 Seattle
9	AKN2627	2016-05-06 09:36:30.0000000	2016-05-06 10:05:00.0000000	-122.35133	47.61527	-122.3343	47.58514	2.22847	69	66	-3	Cedar St 60, 98121 Seattle	1st Ave S 2037, 98134 Seattle
10	AKN2627	2016-05-06 16:34:30.0000000	2016-05-06 17:22:00.0000000	-122.3343	47.58514	-122.29363	47.58763	1.90866	66	63	-3	1st Ave S 2037, 98134 Seattle	30th Ave S 1733, 98144 Seattle

Showing 1 to 10 of 80 entries

Previous 1 2 3 4 5 … 8 Next

DT

- What if you want to filter the data?
  - For example, you want to filter the data based on the Car2Go license plate # and the distance that a vehicle travelled.
  - Do we need to write a new query to retrieve the data from the database?
  - No! We can process the data in the server function.
- We add two selectInputs as the filters in the UI

```
fluidRow(
  column(3,
      selectInput("VID", "Vechile ID:", c("All", unique(as.character(queryData$id))))
  ),
  column(3,
      selectInput("Distance","Distance Larger Than:", c("All", 1, 2, 3, 4, 5))
  )
),
```

Input ID | label | Values to be selected

DT

- In the Server function:

```
# Filter data based on selections
 output$table1 <- DT::renderDataTable(DT::datatable({
   data1 <- queryData
   if (input$VID != "All") {
     data1 <- data1[data1$id == input$VID,]
   }
   if (input$Distance != "All") {
     data1 <- data1[data1$distance >= input$Distance,]
   }
   data1
 }))
```

Filter and update the values in “data1”

Returns the updated “data1”

- Let’s see the UI:

Vechile ID: All

Distance Larger Than: All

Show 10 entries

Search:

	id	otime	dtime	olon	olat	dlon	dlat	distance	ofuel	dfuel	fuel_consumption	oaddress	daddress
1	AKN2620	2016-05-06 09:18:00.0000000	2016-05-06 09:33:30.0000000	-122.38475	47.70117	-122.38721	47.67012	2.14821	100	100	0	NW 99th St 2160, 98117 Seattle	NW 57th St 2283, 98107 Seattle
2	AKN2620	2016-05-06 10:29:30.0000000	2016-05-06 10:57:30.0000000	-122.38721	47.67012	-122.3224	47.60552	5.39222	100	100	0	NW 57th St 2283, 98107 Seattle	Jefferson St 1050, 98104 Seattle
3	AKN2620	2016-05-06 11:49:30.0000000	2016-05-06 12:15:00.0000000	-122.3224	47.60552	-122.32253	47.60457	0.06591	100	100	0	Jefferson St 1050, 98104 Seattle	Terrace St 958, 98104 Seattle
4	AKN2620	2016-05-06 14:12:00.0000000	2016-05-06 14:27:30.0000000	-122.32253	47.60457	-122.33173	47.60259	0.4511	100	100	0	Terrace St 958, 98104 Seattle	James St 271, 98104 Seattle
5	AKN2620	2016-05-06 15:10:30.0000000	2016-05-06 15:34:00.0000000	-122.33173	47.60259	-122.31317	47.55121	3.65411	100	100	0	James St 271, 98104 Seattle	S Orcas St 1527, 98108 Seattle
6	AKN2620	2016-05-06 16:19:30.0000000	2016-05-06 17:56:00.0000000	-122.31317	47.55121	-122.31239	47.55137	0.03812	100	100	0	S Orcas St 1527, 98108 Seattle	16th Ave S 5783, 98108 Seattle
7	AKN2627	2016-05-06 07:37:30.0000000	2016-05-06 08:23:00.0000000	-122.3284	47.68643	-122.34082	47.61089	5.25088	75	69	-6	1st Ave NE 7775, 98115 Seattle	Stewart St 158, 98101 Seattle
8	AKN2627	2016-05-06 08:54:30.0000000	2016-05-06 09:22:30.0000000	-122.34082	47.61089	-122.35133	47.61527	0.57674	69	69	0	Stewart St 158, 98101 Seattle	Cedar St 60, 98121 Seattle
9	AKN2627	2016-05-06 09:36:30.0000000	2016-05-06 10:05:00.0000000	-122.35133	47.61527	-122.3343	47.58514	2.22847	69	66	-3	Cedar St 60, 98121 Seattle	1st Ave S 2037, 98134 Seattle
10	AKN2627	2016-05-06 16:34:30.0000000	2016-05-06 17:22:00.0000000	-122.3343	47.58514	-122.29363	47.58763	1.90866	66	63	-3	1st Ave S 2037, 98134 Seattle	30th Ave S 1733, 98144 Seattle

Showing 1 to 10 of 80 entries

Previous 1 2 3 4 5 … 8 Next

DT

- Please check out the demo source code to see the demos
  - Exercises → Exercise 5 → Scripts → part_2_demo_1_DT.R

- Please check the DT webpage for more functions:
  - https://rstudio.github.io/DT/

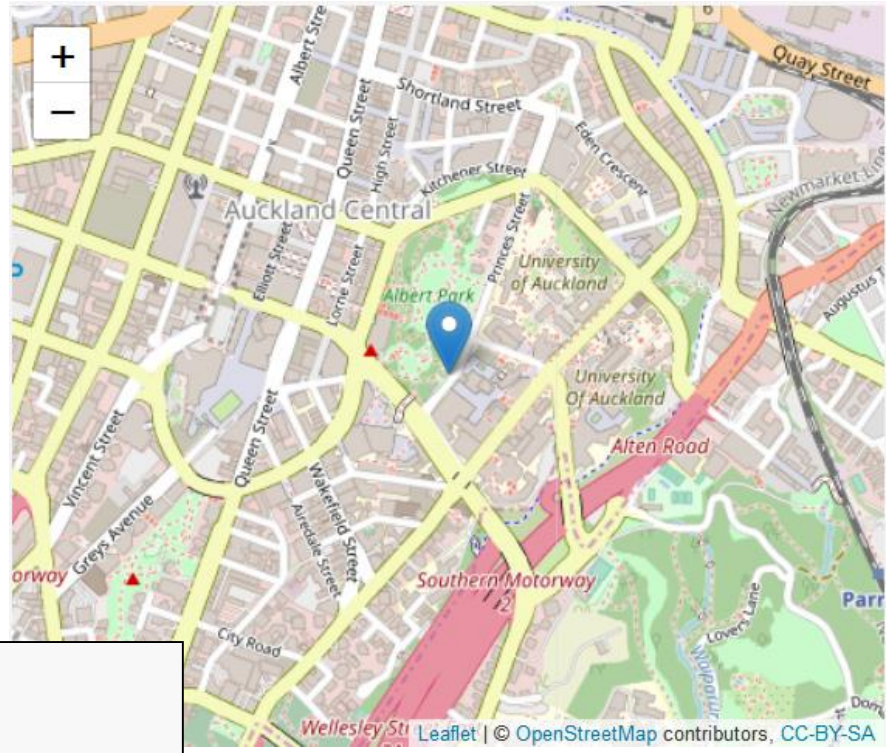
Leaflet

- Leaflet is one of the most popular open-source JavaScript libraries for interactive maps.
  - http://rstudio.github.io/leaflet/
- Add a map layer in your Shiny App
  - In UI:

```
wellPanel(
    leafletOutput("myMap1")
)
```

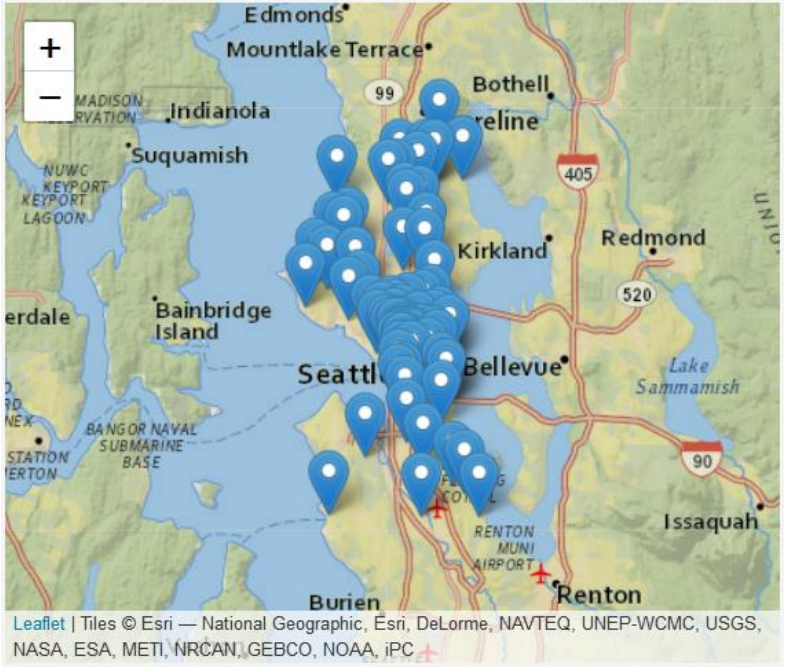
  - In Server function:

```
output$myMap1 <- renderLeaflet({
    leaflet() %>%
      addTiles() %>%  # Add default OpenStreetMap map tiles
      addMarkers(lng=174.768, lat=-36.852, popup="The birthplace of R")
  })
```



Leaflet

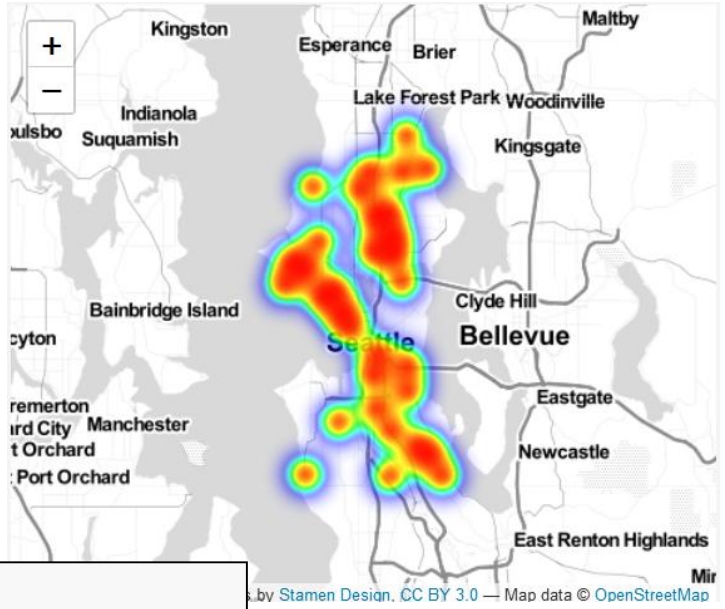
- Add a map layer in your Shiny App
  - You can adjust your map layers
  - Find more layers here:
    - http://leaflet-extras.github.io/leaflet-providers/preview/index.html




```
output$myMap2 <- renderLeaflet({
   leaflet() %>%
     addProviderTiles(providers$Esri.NatGeoWorldMap) %>%
     addMarkers(data = rv$Car2Go, lng = ~olon, lat = ~olat, popup = paste("Vehicle ID:", rv$Car2Go$id, "<br>", "Address:", rv$Car2Go$oaddress))
 })
```

Leaflet

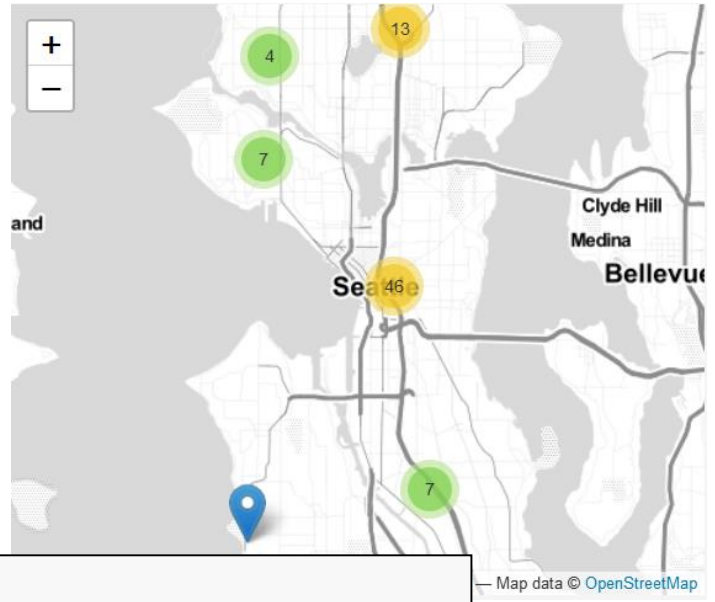
- Add a map layer in your Shiny App
  - You can add your data and visualize them as a heatmap
  - In this example, we visualize the Origins of the trips in the Car2Go data.




```
output$myMap3 <- renderLeaflet({
   leaflet() %>%
     addProviderTiles(providers$Stamen.TonerLite,
                      options = providerTileOptions(noWrap = TRUE)
     ) %>%
     addHeatmap(data = rv$Car2Go, lng = ~olon, lat = ~olat, blur = 20, max = 0.5, radius = 15)
 })
```

Leaflet

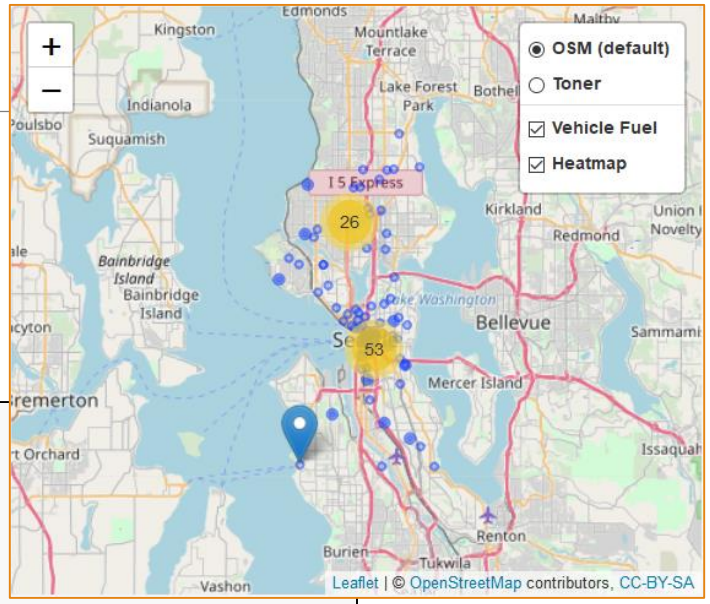
- Add a map layer in your Shiny App
  - You can add your data and visualize them as clusters
  - In this example, we visualize the Origins of the trips in the Car2Go data as clusters.




```
output$myMap4 <- renderLeaflet({
   leaflet() %>%
     addProviderTiles(providers$Stamen.TonerLite,
                      options = providerTileOptions(noWrap = TRUE)
     ) %>%
     addMarkers(data = rv$Car2Go, lng = ~olon, lat = ~olat, clusterOptions = markerClusterOptions())
 })
```

Leaflet

- Add a map layer in your Shiny App
  - You control the layers in leaflet by giving each layer a name (group)




```
output$myMap5 <- renderLeaflet({

  map <- leaflet() %>%
    # Base groups
    addTiles(group = "OSM (default)") %>%
    addProviderTiles(providers$Stamen.Toner, group = "Toner") %>%
    # Overlay groups
    addCircles(data = rv$Car2Go, lng = ~olon, lat = ~olat, radius = rv$Car2Go$ofuel * 2, group = "Vehicle Fuel") %>%
    addMarkers(data = rv$Car2Go, lng = ~olon, lat = ~olat, clusterOptions = markerClusterOptions(), group = "Heatmap") %>%
    # Layers control
    addLayersControl(
      baseGroups = c("OSM (default)", "Toner"),
      overlayGroups = c("Vehicle Fuel", "Heatmap"),
      options = layersControlOptions(collapsed = FALSE)
    )

  map
})
```

Leaflet

- If you want to get the marker's information while you clicking on the marker.

```
output$myMap6 <- renderLeaflet({

  leaflet() %>%
    addProviderTiles(providers$Stamen.TonerLite,
                     options = providerTileOptions(noWrap = TRUE)
    ) %>%
    addMarkers(data = rv$Car2Go, lng = ~olon, lat = ~olat, popup = paste("Vehicle ID:", rv$Car2Go$id, "<br>",
"Address:", rv$Car2Go$oaddress))
 })

 observeEvent(input$myMap6_marker_click, {
   p <- input$myMap6_marker_click
   print(p)

 })
```

- More complicated example shows here(in case you need)
  - https://community.rstudio.com/t/shiny-using-multiple-exclusive-reactivevalues-for-filtering-dataset/40957/5
  - https://stackoverflow.com/questions/51353448/click-on-leaflet-marker-and-get-info

Leaflet

- Please check out the demo source code to see the demos
  - Exercises → Exercise 5 → Scripts → part_2_demo_2_leaflet.R

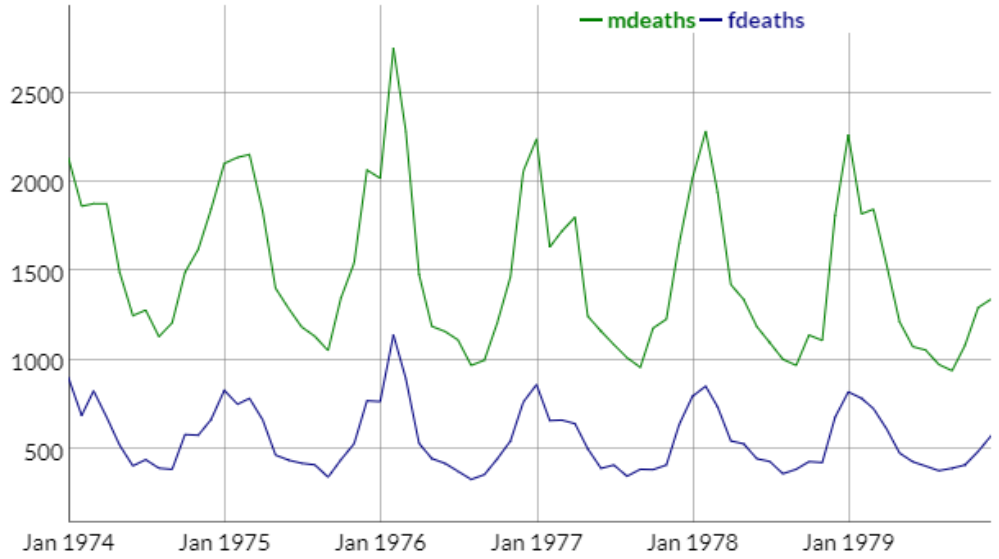
- Please check the Leaflet webpage for more functions:
  - http://rstudio.github.io/leaflet/

dygrpahs for R

- The dygraphs package is an R interface to the dygraphs JavaScript charting library. It provides rich facilities for charting time-series data in R, including:
  - Automatically plots xts time series objects (or any object convertible to xts).
  - Highly configurable axis and series display (including optional second Y-axis).
  - Rich interactive features including zoom/pan and series/point highlighting.
  - Display upper/lower bars (e.g. prediction intervals) around series.
  - Various graph overlays including shaded regions, event lines, and point annotations.
  - Use at the R console just like conventional R plots (via RStudio Viewer).
  - Seamless embedding within R Markdown documents and Shiny web applications.
- http://rstudio.github.io/dygraphs/index.html

dygrpahs for R

- A large portion of transportation related data are time-series data.
- Interactively visualizing time-series data is also very important to find the patterns.
- Hence, we show you a simple demo:



Note that this graph is fully interactive: as your mouse moves over the series individual values are displayed. You can also select regions of the graph to zoom into (double-click zooms out).

dygrpahs for R

```
library(shiny)
library(dygraphs)
library(datasets)

lungDeaths <- cbind(mdeaths, fdeaths)

ui <- fluidPage(
  titlePanel("Predicted Deaths from Lung Disease (UK)"),
  wellPanel(
    dygraphOutput("dygraph")
  )
)

server <- function(input, output) {
  output$dygraph <- renderDygraph({
    dygraph(lungDeaths)
  })
}

shinyApp(ui = ui, server = server)
```

- That all the code needed for run a demo:
  - Exercises → Exercise 5 → Scripts → part_2_demo_3_dygraphs.R
- Dygraphs allows you adjust your UI
- Please check more functions as you nee
  - http://rstudio.github.io/dygraphs/index.html

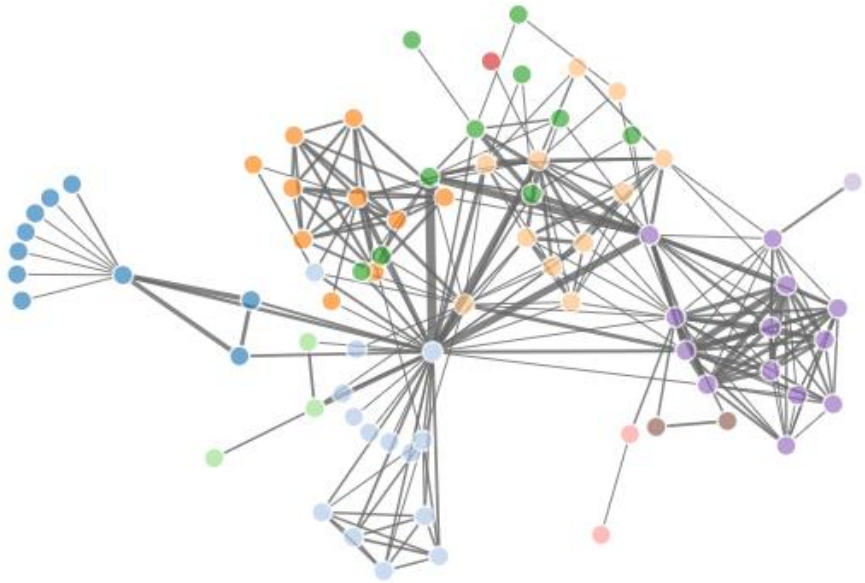
Series Options
Series Highlighting
Axis Options
Labels & Legends
Time Zones
CSS Styling
Range Selector
Candlestick Charts
Synchronization
Straw Broom Charts
Roll Periods
Annotation/Shading
Events and Limits
Upper/Lower Bars
Plugins
Custom Plotters
Colored Ribbon

Other visualization tools

- **networkD3:** D3 JavaScript Network Graphs from R
  - Interactive graph visualization tool
  - Simplified code with no need to write JavaScript
  - Support Shiny
  - Check more functions:
    - http://christophergandrud.github.io/networkD3/

```
# Load data
data(MisLinks)
data(MisNodes)

# Plot
forceNetwork(Links = MisLinks, Nodes = MisNodes,
             Source = "source", Target = "target",
             Value = "value", NodeID = "name",
             Group = "group", opacity = 0.8)
```

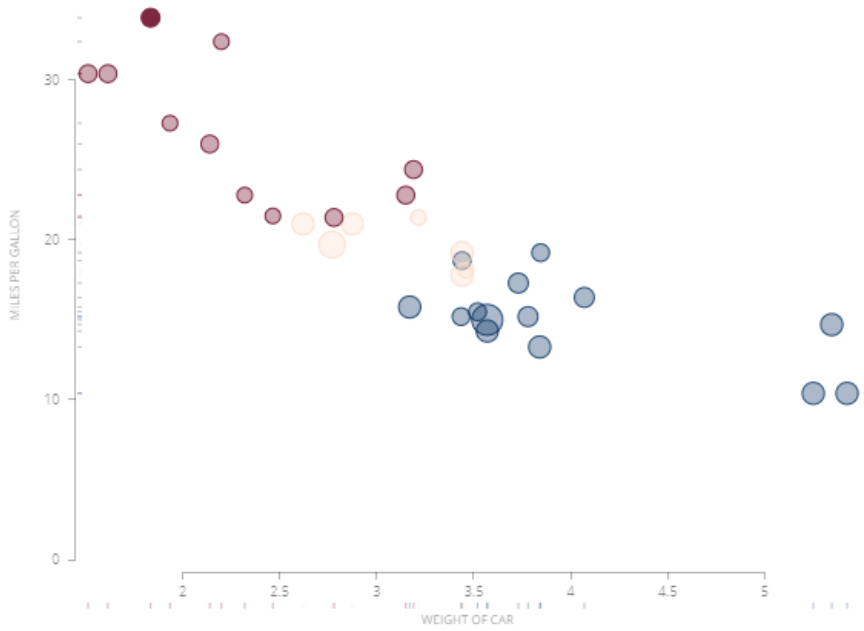


Other visualization tools

- **metricsgraphics**
  - An htmlwidget interface to the MetricsGraphics.js JavaScript/D3 chart library.
  - Interactive charts: line charts, bar charts, scatter plots, multi-line, grids, etc.
  - Check more functions:
    - http://hrbrmstr.github.io/metricsgraphics/

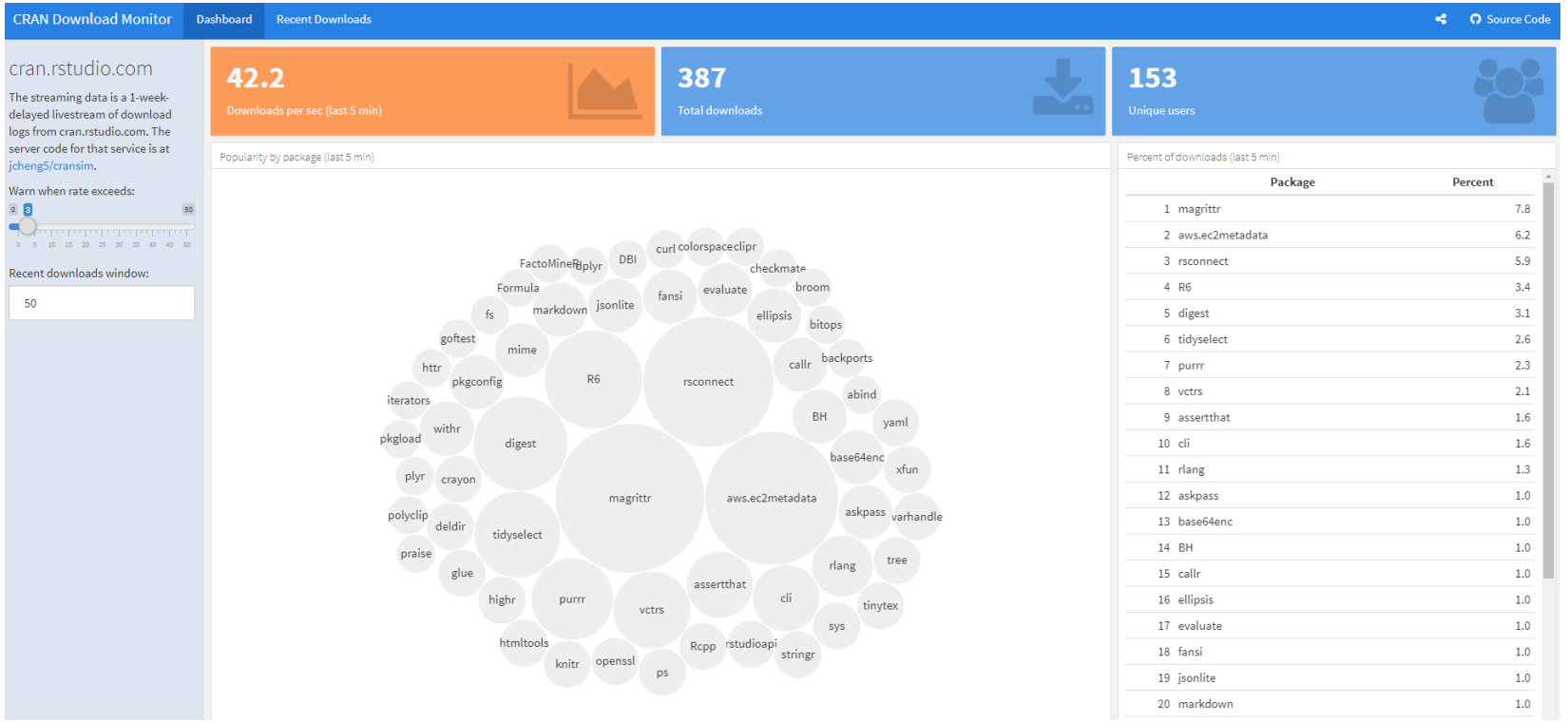
htmlwidgets: HTML Widgets for R

- The **htmlwidgets** package provides a framework for easily creating R bindings to JavaScript libraries
- Packages based on **htmlwidgets:**
  - leaflet -- Interactive maps with OpenStreetMap
  - dygraphs --- Interactive time series visualization
  - networkD3 --- Network visualization with D3
  - sparkline --- Small inline charts
  - DT --- Tabular data via DataTables
  - rthreejs -- Interactive 3D graphics



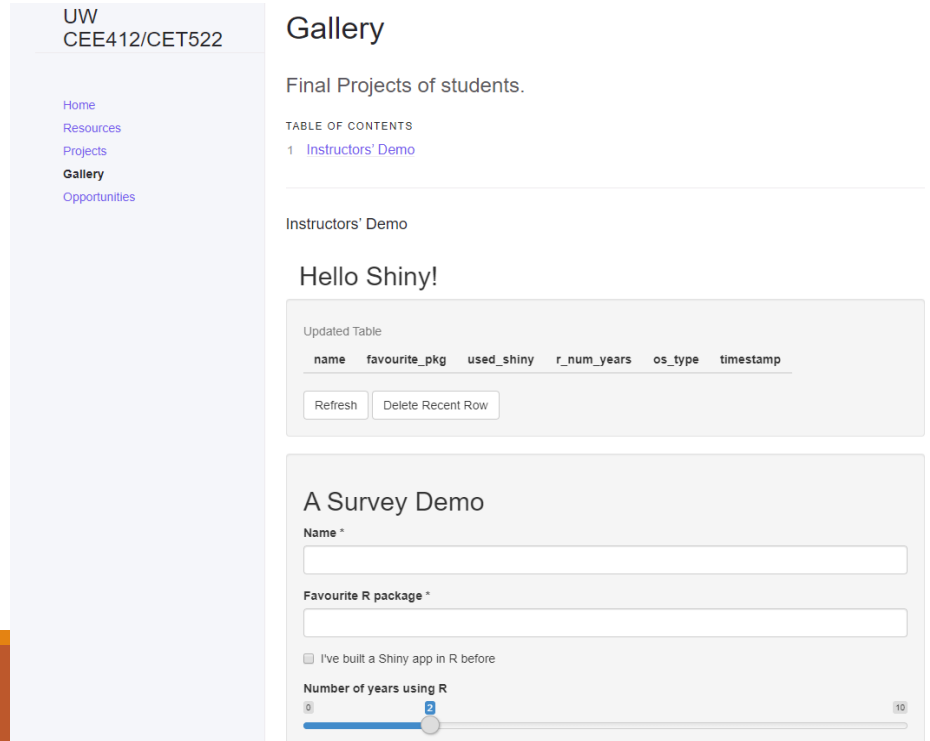
Dashboard

- flexdashboard
  - https://rmarkdown.rstudio.com/flexdashboard/shiny.html
  - An other option to create dashboard other than shinydashboard.



Other Notes

- Shiny App can be integrated in other websites by using the <iframe> tag in HTML.
  - The Shiny App can act as an embedded app in a web page
  - For example, integrate our previous demo in our new course website.
    - https://zhiyongcui.com/CEE412_CET522/docs/gallery/



Summary

- Please try these tools as you need.
- Please try to perfect your Shiny App, especially your general UI.
- The **colors and styles you use in your dashboard** and the **arrangement of the components** will hugely affect your data visualization performance.