(Part A) Build a bike-sharing demand prediction app with R Shiny and Leaflet

Estimated time needed: **90** minutes

Objectives

So far you have built regression models to predict Seoul's hourly bike-sharing demand using weather and date/time information, and the prediction performance is promising.

Now, your team wants to build an interactive **R Shiny dashboard** to be able to visualize weather forecast data and predicted hourly bike-sharing demand for the following cities: `New York, USA`, `Paris, France`, `Suzhou, China`, and `London, UK` (they have similar bike rental fleet sizes as `Seoul, South Korean`).

Within the dashboard, you will build a **Leaflet** based interactive map that shows the max predicted bike-sharing demand in the next 5 days. The predictions will be based on the outcome of a provided regression model (or you can use your own model), which in turn will utilize weather forecast data from the OpenWeather API as input.

This lab requires you to complete the following tasks:

- **TASK 1** : Add a basic max bike prediction overview map
- **TASK 2** : Add a select input (dropdown) to select a specific city

Once you have completed these tasks, you can proceed to next part of the lab, where you will be adding some more plots to your prediction app to display detailed information for a selected city using ggplot. Refer to **Part B- Instructions for enhancing the bike-sharing demand prediction app with City Details Plots**

> Note: For more information on R Shiny package with Leaflet, please refer to Using Leaflet with Shiny

R-Studio on Coursera Labs

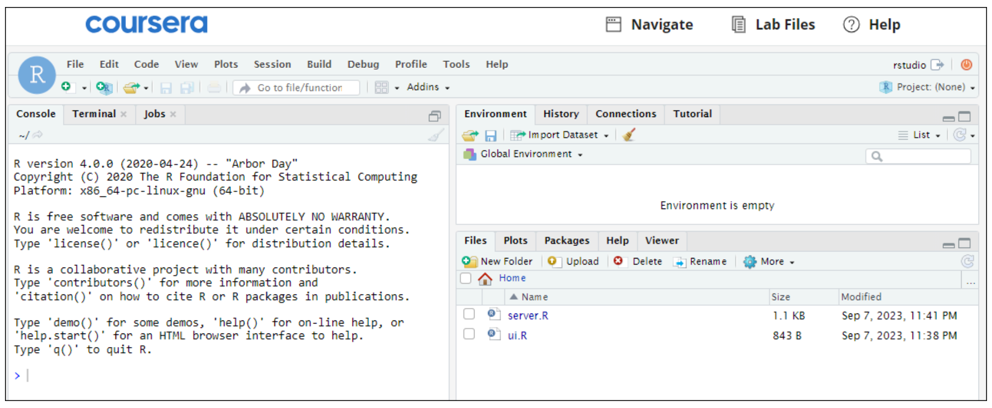
> Note: To complete this lab, you have the option to use either **R-Studio on Coursera Labs** hosted within this course environment or **Posit Cloud Free (earlier RStudio Cloud)**. Both lab environments are cloud based. However, Posit Cloud has some limitations on compute hours in basic service plan.

In this instructions, we guide you through steps required to complete this lab using the **R-Studio on Coursera Labs** hosted within this course environment.

Getting Started

Step 1: Launch RStudio in Coursera

Launch RStudio lab using the `Launch App` button. You will be directed to the RStudio IDE as shown in the screenshot below. Now you are all set to get started on the lab.



```
R version 4.0.0 (2020-04-24) -- "Arbor Day"
Copyright (C) 2020 The R Foundation for Statistical Computing
Platform: x86_64-pc-linux-gnu (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

> 
```

Name	Size	Modified
server.R	1.1 KB	Sep 7, 2023, 11:41 PM
ui.R	843 B	Sep 7, 2023, 11:38 PM

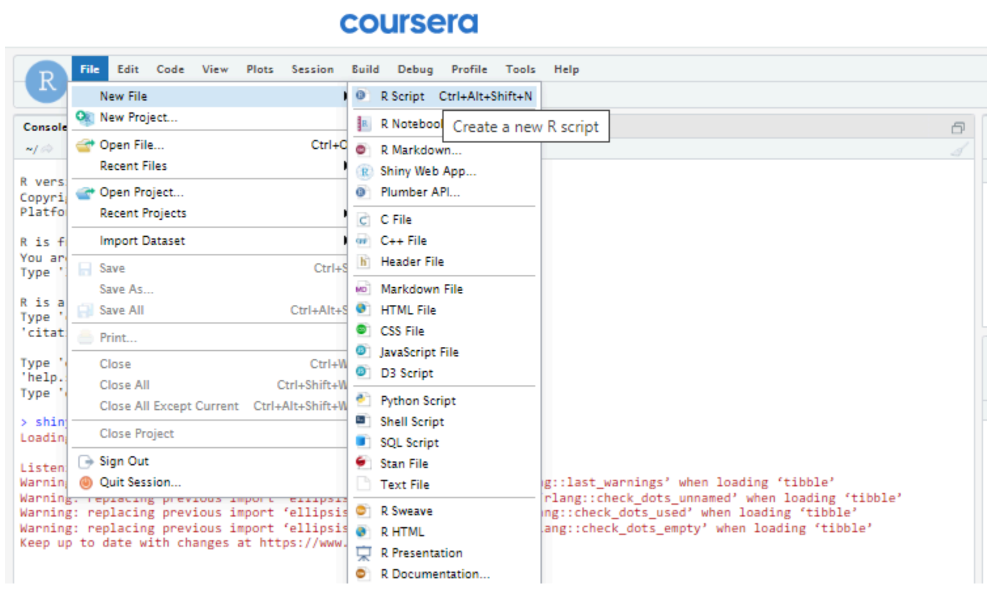
Step 2: Get Familiar with Starter codes (ui.R and Server.R)

Before starting these tasks, you need to become familiar with the UI and Server R scripts. The next section will guide you with this.

As you have learnt before, a Shiny app contains two parts: the **ui.R** and the **Server.R** script files. This lab comes with the pre-loaded starter UI and Server R scripts along with the dataset and model_prediction files within lab environment. These files are available under **`RP0321EN_RCapstone_Week5_Rshiny_StarterCodes`** directory when you launch the lab.

- `ui.R` contains the frontend code of the Shiny app
- `server.R` contains the backend code of the Shiny app
- `selected_cities.csv` file contains geo information about `Seoul, South Korea`, `New York, USA`, `Paris, France`, `Suzhou, China`, and `London, UK`
- `model.csv` file contains a trained regression model with variables, coefficients, and intercept. You may instead use your own model, which you trained in the previous module.
- `model_prediction.R` provides you with functions like OpenWeather API calls, hourly bike-sharing prediction, etc. After executing this script file, you will get a data frame needed to develop the app. If you are interested, you can update the source code and logic in this file to use your own models trained in previous lab.

If these files are not available within the lab, create two R script files in the lab, copy and paste the starter codes given below. Click **File** > **New File** > **R Script** to create new script files as shown in the screenshot below.

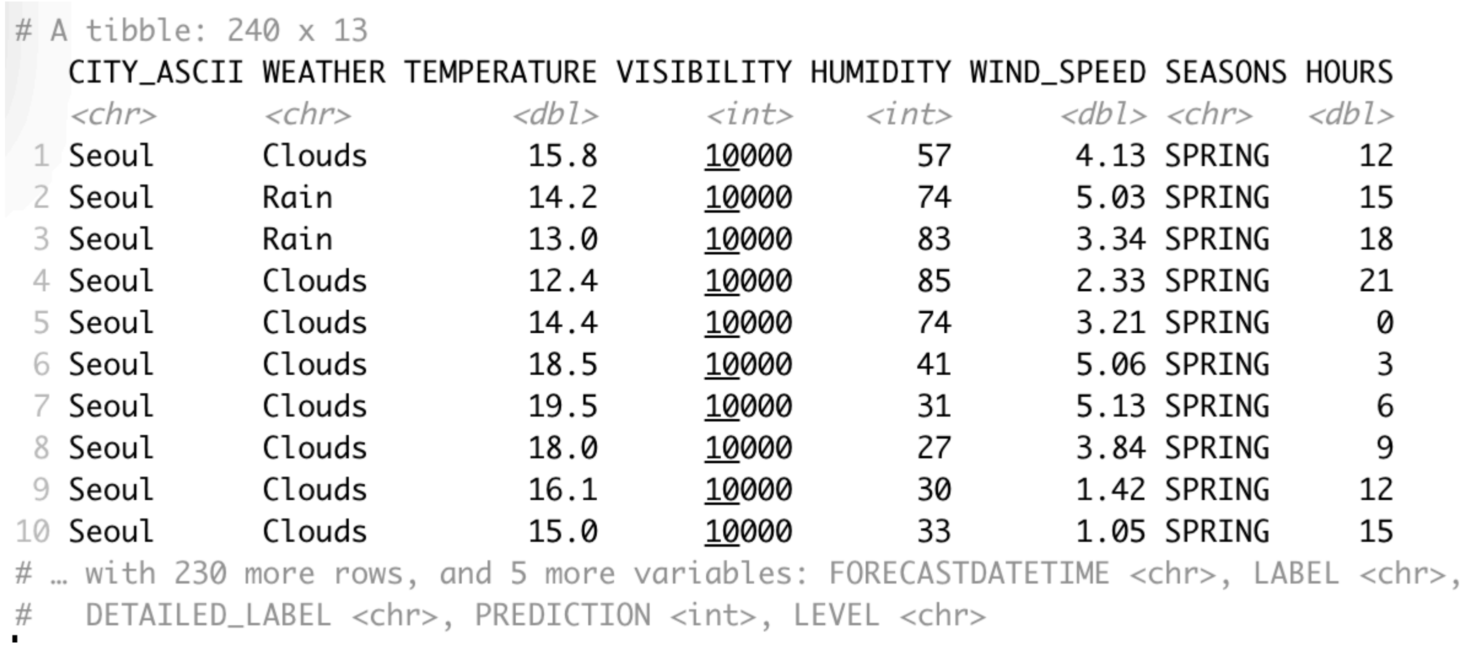


- ▶ Click here for starter code for ui.R file
- ▶ Click here for starter code for server.R file
- ▶ (Optional) Click here to download the included files in this lab

Edit model_prediction.R file

Now, we need to test if the `model_prediction.R` file can generate a proper data frame for you.

- Open `model_prediction.R`, and in Line 24, replace the OpenWeatherAPI key with your own.
- Then open `server.R`, find and uncomment `# test_weather_data_generation()` in `shinyServer` function to run a simple test function. Click `Run App`, and you should see an empty R Shiny app started, and the console should print a data frame called `city_weather_bike_df` which looks like the following one:



```
# A tibble: 240 x 13
   CITY_ASCII WEATHER TEMPERATURE VISIBILITY HUMIDITY WIND_SPEED SEASONS HOURS
   <chr>      <chr>         <dbl>      <int>    <int>      <dbl> <chr>   <dbl>
 1 Seoul      Clouds         15.8      10000       57       4.13 SPRING     12
 2 Seoul      Rain           14.2      10000       74       5.03 SPRING     15
 3 Seoul      Rain           13.0      10000       83       3.34 SPRING     18
 4 Seoul      Clouds         12.4      10000       85       2.33 SPRING     21
 5 Seoul      Clouds         14.4      10000       74       3.21 SPRING      0
 6 Seoul      Clouds         18.5      10000       41       5.06 SPRING      3
 7 Seoul      Clouds         19.5      10000       31       5.13 SPRING      6
 8 Seoul      Clouds         18.0      10000       27       3.84 SPRING      9
 9 Seoul      Clouds         16.1      10000       30       1.42 SPRING     12
10 Seoul      Clouds         15.0      10000       33       1.05 SPRING     15
# … with 230 more rows, and 5 more variables: FORECASTDATETIME <chr>, LABEL <chr>,
#   DETAILED_LABEL <chr>, PREDICTION <int>, LEVEL <chr>
```

This data frame will be your main data source for the R Shiny app, which includes the following columns:

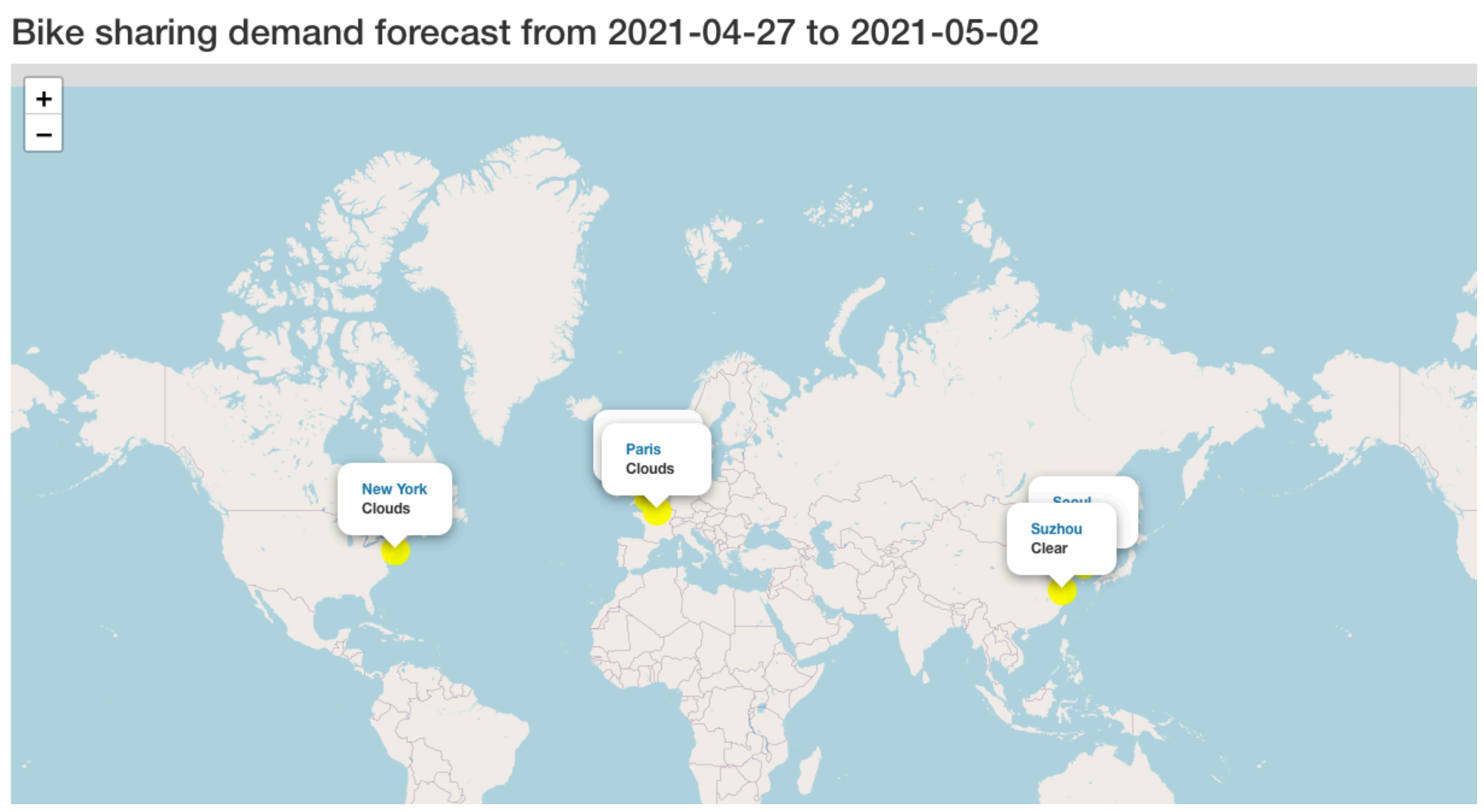
- `CITY_ASCII`: city name in ASCII
- `LNG`: City's Longitude
- `LAT`: City's latitude
- `TEMPERATURE`: The forecasted temperature
- `HUMIDITY`: The forecasted air humidity percentage, unit is `%`
- `BIKE_PREDICTION`: A bike-sharing demand prediction using a regression model, in numeric value
- `BIKE_PREDICTION_LEVEL`: A bike-sharing demand prediction using a regression model, in factor levels (`Small`, `Medium`, `Large`).
- `LABEL`: A weather label in HTML format to be shown on a leaflet marker
- `DETAILED_LABEL`: A detailed weather label in HTML format to be shown on a leaflet popup
- `FORECASTDATETIME`: The detailed forecast date and time

OK, you have made all the preparations and you can start building the R Shiny app.

Task 1: Add a basic max bike prediction overview map

First, from a high-level, we want a map to show which cities may have very large bike-sharing demand in the next 5 days
so we can plan the logistics accordingly (for example, transport some bikes from low-demand cities to high-demand cities).

On the map, we want to add a circle markers with different sizes and colors, and popup labels on each city's location. The color and size of the marker depends on how large the max bike demand prediction is for the next 5 days.
One example of such a map may look like the following:



Bike sharing demand forecast from 2021-04-27 to 2021-05-02

Now, let's add a leaflet output to the UI main panel to create a basic interactive map.

- Open `ui.R`, and add a leaflet output with id `city_bike_map`. You can set other properties such as width or height of the output as well.

Then we need to add backend code to render a leaflet plot with city locations and bike prediction data.

- Open `server.R` and locate the shiny server function within. In this function, you need to create a new data frame called `cities_max_bike` to store the city location and max bike prediction values.
  The new data frame is an aggregated version of `city_weather_bike_df`. You may use `group_by` with any 'finding max' logic to calculate the max bike prediction for each city.

- Then, you need to find and complete the `renderLeaflet()` function to render the required leaflet map.

```

output$city_bike_map <- renderLeaflet({
    # Complete this function to render a leaflet map
  })
```

Copied!

The leaflet map must have the following configurations:

- The aggregated `cities_max_bike` data frame is used as the data source
- There is a circle marker on each city. The color and radius of each circle marker
  depends on the `BIKE_PREDICTION_LEVEL` value in `cities_max_bike`, and has the following parameters:
  - `small`: size 6 and green color
  - `medium`: size 10 and yellow color
  - `large`: size 12 and red color
- There is a popup label for each city, and the label content is the provided `LABEL` value from `cities_max_bike`

Now, run and test your updated app. The rendered leaflet map should look like the example shown earlier.

Task 2: Add a select input (dropdown menu) to select a specific city

OK, you have displayed the max bike prediction for each city. Now you will add an ability to drill down to each city.

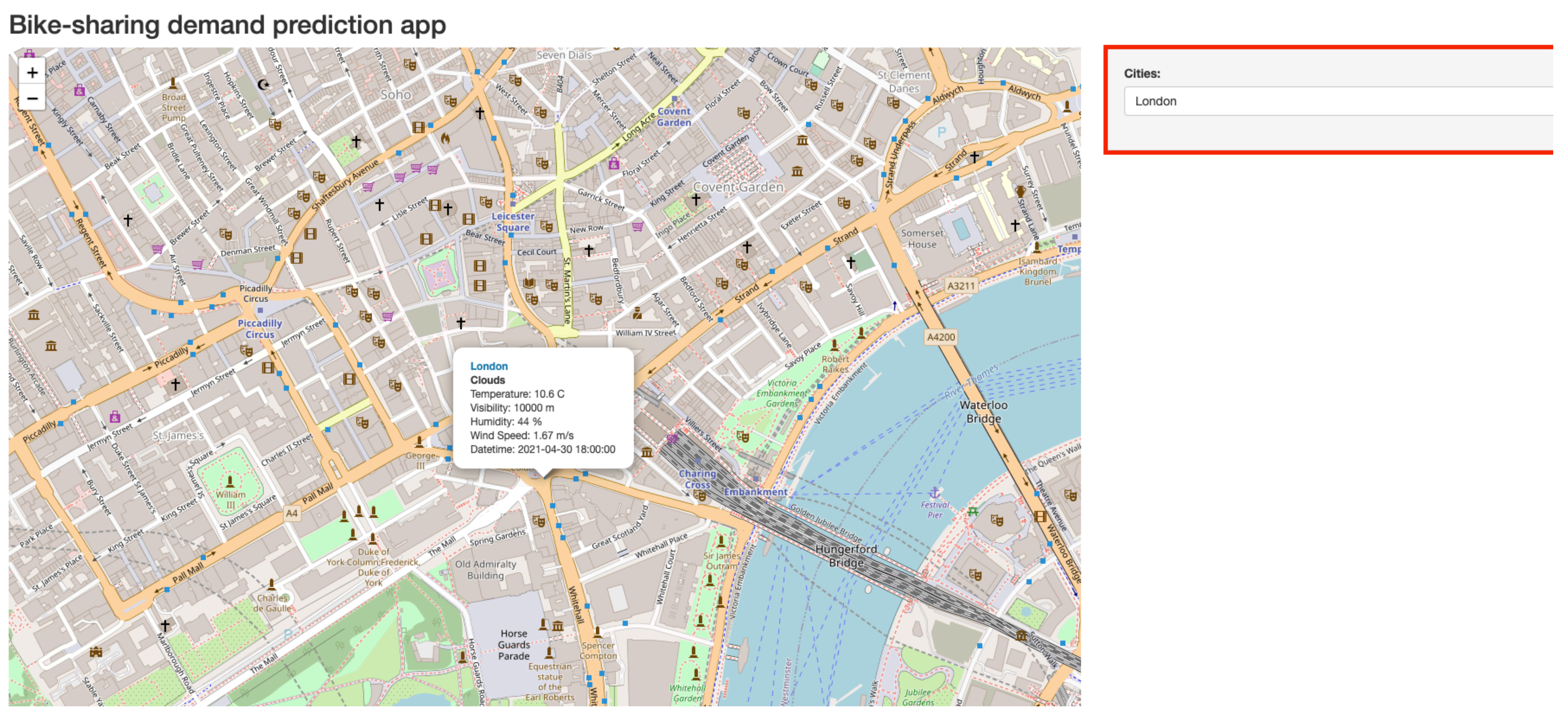
In this task, you can add a 'select input' (a dropdown list) with choices from the following list:

```
c("All", "Seoul", "Suzhou", "London", "New York", "Paris")
```

Copied!

If `All` is selected, then we just show the overview map from the previous task. If any specific city is selected,
you need to drill down to the city, and update its popup label with `DETAILED_LABEL`.

The updated app may look like the following, for example, when `London` is selected:


Bike-sharing demand prediction app

London
Clouds
Temperature: 10.6 C
Visibility: 10000 m
Humidity: 44 %
Wind Speed: 1.67 m/s
Datetime: 2021-04-30 18:00:00

Cities:
London

- Open `ui.R`, and within `sidebarPanel()`, add a 'select input' representing a dropdown list.

```
selectInput(inputId="city_dropdown", ..., )
```

Copied!

Then in the backend server side, you can get the selected value using `input$city_dropdown`.

- Open `server.R`, and wrap the previous `renderLeaflet` code function with:

```
observeEvent(input$city_dropdown, {
   # Execute code when users make selections on the dropdown
})
```

Copied!

Basically, `observeEvent` creates a dropdown selection listener. Whenever you make a selection in the dropdown, the code within its block will be executed.

- Next, within the `observeEvent` function, we need to use an `if` statement to check whether `All` cities or one specific city is selected

```
observeEvent(input$city_dropdown, {
   if(input$city_dropdown != 'All') {
      #Render the city overview map
   }
   else {
      #Render the specific city map
   }
})
```

Copied!

- If a specific city is selected, you need to filter the `cities_max_bike` data frame with the selected city's `CITY_ASCII` name, and make another `renderLeaflet` call to create a new leaflet output with the following configurations:
- Add a simple marker for the selected city
- Add a popup label to each city, where the label content is the provided `DETAILED_LABEL`

Now, test your app with different dropdown selections to make sure it works as designed.

References

You can refer to this link if you need any help with R Shiny with Leaflet:
Using Leaflet with Shiny

Next steps

Great! Now you have added a leaflet output to your R Shiny app which responds to a user's selections.
Once you have completed these tasks, you can proceed to next part of the lab, where you will be adding some more plots to your prediction app to display detailed information for a selected city using `ggplot`. Refer to **Part B- Instructions for enhancing the bike-sharing demand prediction app with City Details Plots**

Author(s)

Yan Luo

Other Contributor(s)

Jeff Grossman

Skills Network