app.R

aem33

2021-12-10

#  
# This is a Shiny web application. You can run the application by clicking  
# the 'Run App' button above.  
#  
# Find out more about building applications with Shiny here:  
#  
# http://shiny.rstudio.com/  
#  
  
library(shiny)  
library(shinyWidgets)  
library(leaflet)  
library(rjson)  
library(sf)

## Linking to GEOS 3.7.1, GDAL 2.4.0, PROJ 5.2.0

library(tidyverse)

## ── Attaching packages ─────────────────────────────────────── tidyverse 1.3.1 ──

## ✓ ggplot2 3.3.5 ✓ purrr 0.3.4  
## ✓ tibble 3.1.5 ✓ dplyr 1.0.7  
## ✓ tidyr 1.1.4 ✓ stringr 1.4.0  
## ✓ readr 2.0.2 ✓ forcats 0.5.1

## ── Conflicts ────────────────────────────────────────── tidyverse\_conflicts() ──  
## x dplyr::filter() masks stats::filter()  
## x dplyr::lag() masks stats::lag()

library(magrittr)

##   
## Attaching package: 'magrittr'

## The following object is masked from 'package:purrr':  
##   
## set\_names

## The following object is masked from 'package:tidyr':  
##   
## extract

library(reticulate)  
library(raster)

## Loading required package: sp

##   
## Attaching package: 'raster'

## The following object is masked from 'package:dplyr':  
##   
## select

library(ncdf4)  
library(lubridate)

##   
## Attaching package: 'lubridate'

## The following objects are masked from 'package:raster':  
##   
## intersect, union

## The following objects are masked from 'package:base':  
##   
## date, intersect, setdiff, union

library(rnaturalearth)  
library(rnaturalearthhires)  
library(plotly)

##   
## Attaching package: 'plotly'

## The following object is masked from 'package:raster':  
##   
## select

## The following object is masked from 'package:ggplot2':  
##   
## last\_plot

## The following object is masked from 'package:stats':  
##   
## filter

## The following object is masked from 'package:graphics':  
##   
## layout

library(rgdal)

## Please note that rgdal will be retired by the end of 2023,  
## plan transition to sf/stars/terra functions using GDAL and PROJ  
## at your earliest convenience.  
##   
## rgdal: version: 1.5-27, (SVN revision 1148)  
## Geospatial Data Abstraction Library extensions to R successfully loaded  
## Loaded GDAL runtime: GDAL 2.4.0, released 2018/12/14  
## Path to GDAL shared files: /usr/share/gdal  
## GDAL binary built with GEOS: TRUE   
## Loaded PROJ runtime: Rel. 5.2.0, September 15th, 2018, [PJ\_VERSION: 520]  
## Path to PROJ shared files: (autodetected)  
## Linking to sp version:1.4-5

library(rgeos)

## rgeos version: 0.5-8, (SVN revision 679)  
## GEOS runtime version: 3.7.1-CAPI-1.11.1   
## Please note that rgeos will be retired by the end of 2023,  
## plan transition to sf functions using GEOS at your earliest convenience.  
## Linking to sp version: 1.4-5   
## Polygon checking: TRUE

# Get CDS CAMS Atmospheric Dataset  
# reticulate::use\_condaenv("r-reticulate")  
# reticulate::py\_run\_file("retrieve\_atm.py")  
  
# Load India and states polygons  
india <- rnaturalearth::ne\_countries(scale = "large", country = "india")  
india\_states <- rnaturalearth::ne\_states(country = "india")  
aqi\_polygon <- sf::st\_as\_sf(india\_states) %>% select(name)  
  
# Read Monthly Raster Data  
pm25 <- raster::brick("cams\_atmosphereData.nc", varname = "pm2p5") %>% mask(india)  
pm10 <- raster::brick("cams\_atmosphereData.nc", varname = "pm10") %>% mask(india)  
co <- raster::brick("cams\_atmosphereData.nc", varname = "tcco") %>% mask(india)  
no2 <- raster::brick("cams\_atmosphereData.nc", varname = "tcno2") %>% mask(india)  
so2 <- raster::brick("cams\_atmosphereData.nc", varname = "tcso2") %>% mask(india)  
o3 <- raster::brick("cams\_atmosphereData.nc", varname = "gtco3") %>% mask(india)  
  
# Create List of Pollutants  
polList <- list(pm25, pm10, co, no2, so2, o3)  
names(polList) <- c("pm25", "pm10", "co", "no2", "so2", "o3")  
  
# Create Time Series Table  
get\_timeSeries <- function(date\_start, date\_end, area){  
   
 if (area != "All") {  
 maskArea <- subset(india\_states, india\_states$name == area)  
 ts\_pm25 <- mask(pm25, maskArea)  
 ts\_pm10 <- mask(pm10, maskArea)  
 ts\_co <- mask(co, maskArea)  
 ts\_no2 <- mask(no2, maskArea)  
 ts\_so2 <- mask(so2, maskArea)  
 ts\_o3 <- mask(o3, maskArea)  
   
 } else {  
 ts\_pm25 <- pm25  
 ts\_pm10 <- pm10  
 ts\_co <- co  
 ts\_no2 <- no2  
 ts\_so2 <- so2  
 ts\_o3 <- o3  
 }  
   
 polTS <- tibble(  
 id = names(co),  
 date = as.Date(id, "X%Y.%m.%d"),  
 pm25 = NA,  
 pm10 = NA,  
 co = NA,  
 no2 = NA,  
 so2 = NA,  
 o3 = NA  
 ) %>%   
 column\_to\_rownames("id")  
   
 for (dates in names(co)) {  
 polTS[dates, "pm25"] <- subset(ts\_pm25, dates) %>% values() %>% mean(na.rm = TRUE) \* 1e9  
 polTS[dates, "pm10"] <- subset(ts\_pm10, dates) %>% values() %>% mean(na.rm = TRUE) \* 1e9  
 polTS[dates, "co"] <- subset(ts\_co, dates) %>% values() %>% mean(na.rm = TRUE) \* 1e5  
 polTS[dates, "no2"] <- subset(ts\_no2, dates) %>% values() %>% mean(na.rm = TRUE) \* 1e5  
 polTS[dates, "so2"] <- subset(ts\_so2, dates) %>% values() %>% mean(na.rm = TRUE) \* 1e5  
 polTS[dates, "o3"] <- subset(ts\_o3, dates) %>% values() %>% mean(na.rm = TRUE) \* 1e5  
 }  
   
 polTS %>%   
 filter(date >= date\_start & date <= date\_end) %>%   
 as\_tibble()  
}  
  
# Choices for Pollutants  
pollutants <- c("pm25", "pm10", "co", "no2", "so2", "o3")  
pol\_vars <- c(  
 "PM2.5" = "pm25",  
 "PM10" = "pm10",  
 "CO" = "co",  
 "NO2" = "no2",  
 "SO2" = "so2",  
 "Ozone" = "o3"  
)  
pol\_aqi <- c(  
 "PM2.5" = "pm25",  
 "PM10" = "pm10"  
)  
  
# Choices for Indian States  
source("india\_state.R", echo = FALSE)  
  
# Create Forecast Table  
source("forecast.R", echo = FALSE)  
  
# Define UI for application that shows the map  
ui <- shinyUI(  
 navbarPage(  
 "Project 17",  
 tabPanel(  
 "Monthly Mean CAMS Map",  
 div(  
 class = "outer",  
 tags$head(  
 includeCSS("style.css"),  
 tags$style(  
 "#controls\_monthly {  
 overflow: auto;  
 }"  
 )  
 ),  
 leafletOutput("map\_monthly", width = "100%", height = "100%"),  
 absolutePanel(  
 id = "AQI\_Table\_monthly",  
 class = "panel panel-default",  
 fixed = TRUE,  
 draggable = FALSE,  
 top = 60,  
 left = 20,  
 right = "auto",  
 bottom = "auto",  
 width = 330,  
 height = "auto",  
   
 h2("Air Quality Index (AQI)"),  
   
 img(src = "AQI\_Table.png", width = "100%"),  
   
 h4("Air Quality Index were derived from parameters provided by Indian Central Pollution Board Control")  
 ),  
 absolutePanel(  
 id = "controls\_monthly",  
 class = "panel panel-default",  
 fixed = TRUE,  
 draggable = TRUE,  
 top = 60,  
 left = "auto",  
 right = 20,  
 bottom = "auto",  
 width = 330,  
 height = "auto",  
   
 h2("Parameter Settings"),  
   
 switchInput(inputId = "aqi\_monthly", value = FALSE, label = "Show AQI"),  
   
 conditionalPanel(  
 "input.aqi\_monthly == true",  
 selectInput("pollutant\_monthly\_aqi", "Pollutant (AQI)", pol\_aqi)  
 ),  
   
 conditionalPanel(  
 "input.aqi\_monthly == false",  
 selectInput("pollutant\_monthly", "Pollutant", pol\_vars)  
 ),  
   
 airDatepickerInput(  
 "date\_monthly",  
 "Date:",  
 value = "2015-01-01",  
 minDate = "2015-01-01",  
 maxDate = "2020-12-01",  
 view = "months",  
 minView = "months",  
 dateFormat = "yyyy-mm"  
 ),  
   
 selectInput("state\_month", "State", states\_choices),  
   
 uiOutput("mt\_stateName"),  
   
 plotlyOutput("monthlyTS", height = "auto"),  
   
 h4("Contains modified Copernicus Atmosphere Monitoring Service Information")  
 )  
 )  
 ),  
 tabPanel(  
 "Forecast CAMS Map",  
 div(  
 class = "outer",  
 tags$head(  
 includeCSS("style.css")  
 ),  
 leafletOutput("map\_forecast", width = "100%", height = "100%"),  
 absolutePanel(  
 id = "AQI\_Table\_forecast",  
 class = "panel panel-default",  
 fixed = TRUE,  
 draggable = TRUE,  
 top = 60,  
 left = 20,  
 right = "auto",  
 bottom = "auto",  
 width = 330,  
 height = "auto",  
   
 h2("Air Quality Index (AQI)"),  
   
 img(src = "AQI\_Table.png", width = "100%"),  
   
 h4("Air Quality Index were derived from parameters provided by Indian Central Pollution Board Control")  
 ),  
 absolutePanel(  
 id = "controls\_forecast",  
 class = "panel panel-default",  
 fixed = TRUE,  
 draggable = FALSE,  
 top = 60,  
 left = "auto",  
 right = 20,  
 bottom = "auto",  
 width = 330,   
 height = "auto",  
   
 h2("Parameter Settings"),  
   
 switchInput(inputId = "aqi\_forecast", value = FALSE, label = "Show AQI"),  
   
 conditionalPanel(  
 "input.aqi\_forecast == true",  
 selectInput("pollutant\_forecast\_aqi", "Pollutant (AQI)", pol\_aqi)  
 ),  
   
 conditionalPanel(  
 "input.aqi\_forecast == false",  
 selectInput("pollutant\_forecast", "Pollutant", pol\_vars)  
 ),  
   
 airDatepickerInput(  
 "date\_forecast",  
 "Date:",  
 value = min(names(forecastTable$pm25)) %>% as.Date(format = "X%Y.%m.%d"),  
 minDate = min(names(forecastTable$pm25)) %>% as.Date(format = "X%Y.%m.%d"),  
 maxDate = max(names(forecastTable$pm25)) %>% as.Date(format = "X%Y.%m.%d"),  
 view = "days",  
 minView = "days",  
 dateFormat = "dd-mm-yyyy"  
 ),  
   
 selectInput("state\_fcast", "State", states\_choices),  
   
 uiOutput("fc\_stateName"),  
   
 plotlyOutput("forecastTS", height = "auto"),  
   
 h4("Contains modified Copernicus Atmosphere Monitoring Service Information")  
 )  
 )  
 )  
 )  
)  
  
# Define server logic required to draw a histogram  
server <- function(input, output) {  
   
 # Create Leaflet Map  
 output$map\_monthly <- renderLeaflet({  
 leaflet(india\_states, options = leafletOptions(zoomControl = FALSE)) %>%  
 addProviderTiles(providers$Stamen.Toner) %>%   
 setView(lng = 77, lat = 22, zoom = 5) %>%   
 addRasterImage(pm25$X2015.01.01, colors = "viridis")  
 })  
   
 output$map\_forecast <- renderLeaflet({  
 leaflet(data = india\_states, leafletOptions(zoomControl = FALSE)) %>%  
 addProviderTiles(providers$Stamen.Toner) %>%   
 setView(lng = 77, lat = 22, zoom = 5) %>%   
 addRasterImage(forecastTable$pm25[[names(forecastTable$pm25)[1]]], colors = "viridis")  
 })  
   
 # Create Time Series Plot  
   
 output$monthlyTS <- renderPlotly({  
 TS\_Table <- get\_timeSeries("2015-01-01", "2020-12-01", "All")  
   
 mTS <- TS\_Table %>%   
 pivot\_longer(cols = -date, names\_to = "pol", values\_to = "value") %>%   
 mutate(pol = fct\_relevel(pol, c("pm25", "pm10", "co", "no2", "so2", "o3"))) %>%   
 ggplot() +  
 geom\_line(aes(x = date, y = value, color = pol), show.legend = FALSE) +  
 facet\_wrap(  
 pol ~ .,  
 ncol = 1,  
 scales = "free\_y",  
 labeller = labeller(  
 pol = c(  
 "pm25" = "PM 2.5",  
 "pm10" = "PM 10",  
 "co" = "Carbon Monoxide (CO)",  
 "no2" = "Nitrogen Dioxide (NO2)",  
 "so2" = "Sulphur Dioxide (SO2)",  
 "o3" = "Ozone"  
 ))  
 ) + labs(x = "Date", y = "") + theme(legend.position = "none")  
   
 ggplotly(mTS)  
 })  
   
 output$forecastTS <- renderPlotly({  
 TS\_Table <- fc\_timeSeries("All")  
   
 mTS <- TS\_Table %>%   
 pivot\_longer(cols = -date, names\_to = "pol", values\_to = "value") %>%   
 mutate(pol = fct\_relevel(pol, c("pm25", "pm10", "co", "no2", "so2", "o3"))) %>%   
 ggplot() +  
 geom\_line(aes(x = date, y = value, color = pol), show.legend = FALSE) +  
 facet\_wrap(  
 pol ~ .,  
 ncol = 1,  
 scales = "free\_y",  
 labeller = labeller(  
 pol = c(  
 "pm25" = "PM 2.5",  
 "pm10" = "PM 10",  
 "co" = "Carbon Monoxide (CO)",  
 "no2" = "Nitrogen Dioxide (NO2)",  
 "so2" = "Sulphur Dioxide (SO2)",  
 "o3" = "Ozone"  
 ))  
 ) + labs(x = "Date", y = "") + theme(legend.position = "none")  
   
 ggplotly(mTS)  
 })  
   
 # Observer for pollutant concentration input  
   
 observe({  
 ptn\_month <- input$pollutant\_monthly  
 ptn\_month\_aqi <- input$pollutant\_monthly\_aqi  
 ptn\_fcast <- input$pollutant\_forecast  
 ptn\_fcast\_aqi <- input$pollutant\_forecast\_aqi  
   
 polRange <- tibble(  
 pollutant = c("pm25", "pm10", "co", "no2", "so2", "o3"),  
 btm\_range = c(0, 0, 20, 0, 0, 400),  
 upr\_range = c(600, 800, 300, 2, 3, 800)  
 )  
   
 pal\_month <- colorNumeric(  
 "viridis",   
 c(  
 polRange[polRange$pollutant == ptn\_month, "btm\_range"] %>% as.numeric(),  
 polRange[polRange$pollutant == ptn\_month, "upr\_range"] %>% as.numeric()  
 ),   
 na.color = rgb(0, 0, 0, 0))  
   
 pal\_fcast <- colorNumeric(  
 "viridis",   
 c(  
 polRange[polRange$pollutant == ptn\_fcast, "btm\_range"] %>% as.numeric(),  
 polRange[polRange$pollutant == ptn\_fcast, "upr\_range"] %>% as.numeric()  
 ),   
 na.color = rgb(0, 0, 0, 0))  
   
 if (!input$aqi\_monthly){  
   
 # Monthly Plot  
 mapdate <- input$date\_monthly  
 mapdate <- paste0("X", format(mapdate, "%Y.%m.%d"))  
   
 mapRast\_month <- polList[[ptn\_month]] %>% subset(mapdate)  
 colorData <- values(mapRast\_month)  
   
 leafletProxy("map\_monthly") %>%   
 clearShapes() %>%   
 clearImages() %>%   
 addRasterImage(mapRast\_month, colors = "viridis") %>%   
 addLegend(  
 "bottomleft",   
 opacity = 1.0,  
 pal = pal\_month,   
 values = c(  
 polRange[polRange$pollutant == ptn\_month, "btm\_range"] %>% as.numeric(),  
 polRange[polRange$pollutant == ptn\_month, "upr\_range"] %>% as.numeric()  
 ),   
 title = ifelse(  
 ptn\_month %in% c("pm25", "pm10"),  
 paste(str\_to\_upper(ptn\_month), "in µg/m^3"),  
 paste(str\_to\_upper(ptn\_month), "in µg/cm^2")  
 ),  
 layerId = "colorLegend",  
 na.label = ""  
 )  
  
 } else {  
   
 # Monthly Plot  
 mapdate <- input$date\_monthly  
 mapdate <- paste0("X", format(mapdate, "%Y.%m.%d"))  
   
 mapRast\_month <- polList[[ptn\_month\_aqi]] %>% subset(mapdate)  
 colorData <- values(mapRast\_month)  
   
 aqiTable <- aqi\_polygon %>%  
 mutate(  
 value = extract(mapRast\_month, aqi\_polygon) %>% lapply(mean) %>% unlist()  
 )  
   
 bins <- c(0, 50, 100, 200, 300, 400, 800)  
 pal <- colorBin(c("#00b050", "#92d050", "#ffff00", "#ff9900", "#ff0000", "#c00000"), domain = aqiTable$value, bins = bins)  
   
 leafletProxy("map\_monthly", data = aqiTable) %>%   
 clearImages() %>%   
 addPolygons(  
 fillColor = ~pal(value \* 1e9),  
 fillOpacity = 1.0,  
 opacity = 1.0,  
 color = "black",  
 weight = 1.0,  
 )  
  
 }  
   
 if (!input$aqi\_forecast) {  
 # Forecast Plot  
 mapdate <- input$date\_forecast  
 mapdate <- paste0("X", format(mapdate, "%Y.%m.%d"))  
   
 mapRast\_forecast <- forecastTable[[ptn\_fcast]] %>% subset(mapdate)  
 colorData <- values(mapRast\_forecast)  
   
 leafletProxy("map\_forecast") %>%   
 clearImages() %>%  
 clearShapes() %>%   
 addRasterImage(mapRast\_forecast, colors = "viridis") %>%   
 addLegend(  
 "bottomleft",   
 pal = pal\_fcast,  
 opacity = 1.0,  
 values = c(  
 polRange[polRange$pollutant == ptn\_fcast, "btm\_range"] %>% as.numeric(),  
 polRange[polRange$pollutant == ptn\_fcast, "upr\_range"] %>% as.numeric()  
 ),   
 title = ifelse(  
 ptn\_month %in% c("pm25", "pm10"),  
 paste(str\_to\_upper(ptn\_fcast), "in µg/m^3"),  
 paste(str\_to\_upper(ptn\_fcast), "in µg/cm^2")  
 ),   
 layerId = "colorLegend",  
 na.label = "",  
 )  
 } else {  
 # Forecast Plot  
 mapdate <- input$date\_forecast  
 mapdate <- paste0("X", format(mapdate, "%Y.%m.%d"))  
   
 mapRast\_fcast <- forecastTable[[ptn\_fcast\_aqi]] %>% subset(mapdate)  
 colorData <- values(mapRast\_fcast)  
   
 aqiTable <- aqi\_polygon %>%  
 mutate(  
 value = extract(mapRast\_fcast, aqi\_polygon) %>% lapply(mean) %>% unlist()  
 )  
   
 bins <- c(0, 50, 100, 200, 300, 400, 800)  
 pal <- colorBin(c("#00b050", "#92d050", "#ffff00", "#ff9900", "#ff0000", "#c00000"), domain = aqiTable$value, bins = bins)  
   
 leafletProxy("map\_forecast", data = aqiTable) %>%  
 clearImages() %>%  
 addPolygons(  
 fillColor = ~pal(value \* 1e9),  
 fillOpacity = 1.0,  
 opacity = 1.0,  
 color = "black",  
 weight = 1.0,  
 )  
 }  
   
 })  
   
 # Observe for state input in Raster Map  
  
 observe({  
 area\_month <- input$state\_month  
 area\_fcast <- input$state\_fcast  
   
 # Change the text in Time Series  
 output$mt\_stateName <- renderUI({  
 h4(id = "mt\_stateName", paste("Time Series of", ifelse(area\_month != "All", area\_month, "All India")))  
 })  
   
 output$fc\_stateName <- renderUI({  
 h4(id = "fc\_stateName", paste("Time Series of", ifelse(area\_fcast != "All", area\_fcast, "All India")))  
 })  
   
 ###### MAP  
 # Monthly Map  
 if (area\_month != "All") {  
 area\_pol\_month <- subset(india\_states, india\_states$name == area\_month)  
 area\_ext\_month <- area\_pol\_month %>% extent()  
   
 leafletProxy("map\_monthly", data = area\_pol\_month) %>%  
 removeShape("selectedState") %>%  
 flyTo(  
 lng = mean(c(area\_ext\_month[1], area\_ext\_month[2])),  
 lat = mean(c(area\_ext\_month[3], area\_ext\_month[4])),  
 zoom = 6  
 ) %>%  
 addPolygons(  
 layerId = "selectedState",  
 color = "red",  
 opacity = 1.0,  
 fillOpacity = 0.05  
 )  
  
 } else {  
 leafletProxy("map\_monthly") %>%  
 removeShape("selectedState") %>%  
 flyTo(lng = 77, lat = 22, zoom = 5)  
 }  
   
 # Forecast Map  
 if (area\_fcast != "All") {  
 area\_pol\_fcast <- subset(india\_states, india\_states$name == area\_fcast)  
   
 area\_ext\_fcast <- area\_pol\_fcast %>% extent()  
   
 leafletProxy("map\_forecast", data = area\_pol\_fcast) %>%  
 removeShape("selectedState") %>%  
 flyTo(  
 lng = mean(c(area\_ext\_fcast[1], area\_ext\_fcast[2])),  
 lat = mean(c(area\_ext\_fcast[3], area\_ext\_fcast[4])),  
 zoom = 6  
 ) %>%  
 addPolygons(  
 layerId = "selectedState",  
 color = "red",  
 opacity = 1.0,  
 fillOpacity = 0.05  
 )  
   
 } else {  
 leafletProxy("map\_forecast") %>%  
 removeShape("selectedState") %>%  
 flyTo(lng = 77, lat = 22, zoom = 5)  
 }  
   
 ###### TIME - SERIES  
   
 output$monthlyTS <- renderPlotly({  
 TS\_Table <- get\_timeSeries("2015-01-01", "2020-12-01", area\_month)  
   
 mTS <- TS\_Table %>%   
 pivot\_longer(cols = -date, names\_to = "pol", values\_to = "value") %>%   
 mutate(pol = fct\_relevel(pol, c("pm25", "pm10", "co", "no2", "so2", "o3"))) %>%   
 ggplot() +  
 geom\_line(aes(x = date, y = value, color = pol), show.legend = FALSE) +  
 facet\_wrap(  
 pol ~ .,  
 ncol = 1,  
 scales = "free\_y",  
 labeller = labeller(  
 pol = c(  
 "pm25" = "PM 2.5",  
 "pm10" = "PM 10",  
 "co" = "Carbon Monoxide (CO)",  
 "no2" = "Nitrogen Dioxide (NO2)",  
 "so2" = "Sulphur Dioxide (SO2)",  
 "o3" = "Ozone"  
 ))  
 ) + labs(x = "Date", y = "") + theme(legend.position = "none")  
   
 ggplotly(mTS)  
 })  
   
 # Forecast TS  
   
 output$forecastTS <- renderPlotly({  
 TS\_Table <- fc\_timeSeries(area\_fcast)  
   
 mTS <- TS\_Table %>%   
 pivot\_longer(cols = -date, names\_to = "pol", values\_to = "value") %>%   
 mutate(pol = fct\_relevel(pol, c("pm25", "pm10", "co", "no2", "so2", "o3"))) %>%   
 ggplot() +  
 geom\_line(aes(x = date, y = value, color = pol), show.legend = FALSE) +  
 facet\_wrap(  
 pol ~ .,  
 ncol = 1,  
 scales = "free\_y",  
 labeller = labeller(  
 pol = c(  
 "pm25" = "PM 2.5",  
 "pm10" = "PM 10",  
 "co" = "Carbon Monoxide (CO)",  
 "no2" = "Nitrogen Dioxide (NO2)",  
 "so2" = "Sulphur Dioxide (SO2)",  
 "o3" = "Ozone"  
 ))  
 ) + labs(x = "Date", y = "") + theme(legend.position = "none")  
   
 ggplotly(mTS)  
 })  
 })  
}  
  
# Run the application   
shinyApp(ui = ui, server = server)

## PhantomJS not found. You can install it with webshot::install\_phantomjs(). If it is installed, please make sure the phantomjs executable can be found via the PATH variable.