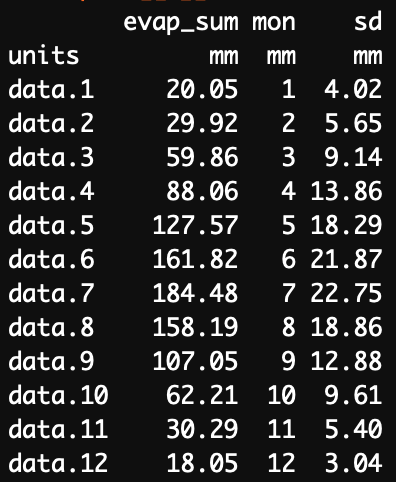
output1 <- OptimalStation(40.762428, -111.860265, '2017-01-01', '2018-01-01')



works completely: all outputs are yielded

ETo format:

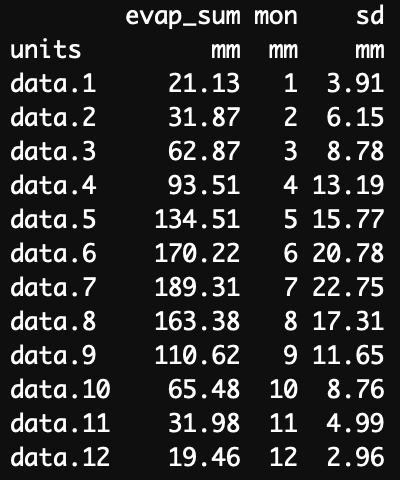


output2 <- OptimalStation(40.692409, -111.835631, '2017-01-01', '2018-01-01')



works completely: all outputs are yielded

ETo format:

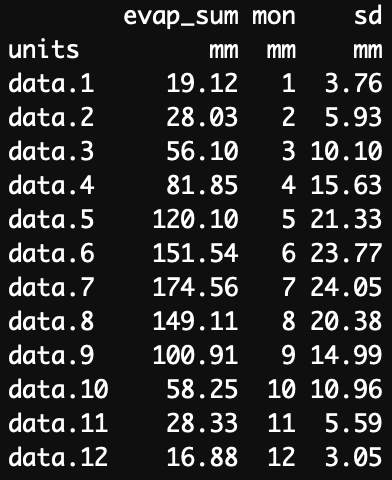


output3 <- OptimalStation(40.887758, -111.877941, '2017-01-01', '2018-01-01')



works completely: all outputs are yielded

ETo format:

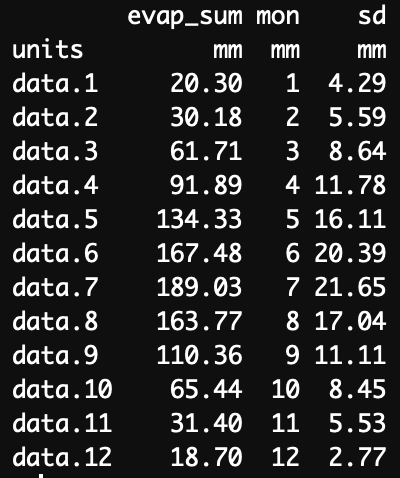


output4 <- OptimalStation(40.690215, -112.008400, '2017-01-01', '2018-01-01')



works completely: all outputs are yielded

ETo format:

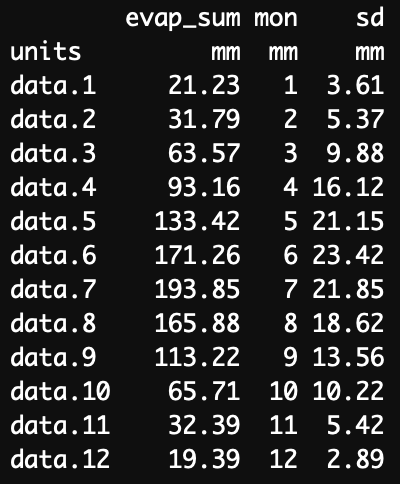


output5 <- OptimalStation(40.386067, -111.801127, '2017-01-01', '2018-01-01')



works completely: all outputs are yielded

ETo format:



#

# 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/

#

#EXAMPLE for calculating with user-input values

#https://stackoverflow.com/questions/40997817/reactive-variables-in-shiny-for-later-calculations

######GLOBAL####GLOBAL####GLOBAL####GLOBAL####GLOBAL####GLOBAL####GLOBAL####GLOBAL####GLOBAL####GLOBAL####

######GLOBAL####GLOBAL####GLOBAL####GLOBAL####GLOBAL####GLOBAL####GLOBAL####GLOBAL####GLOBAL####GLOBAL####

######GLOBAL####GLOBAL####GLOBAL####GLOBAL####GLOBAL####GLOBAL####GLOBAL####GLOBAL####GLOBAL####GLOBAL####

library(shiny)

library(leaflet)

library(rnoaa)

library(ggmap)

library(ggplot2)

library(dplyr)

library(rjson)

library(jsonlite)

library(RCurl)

crops <- read.csv('data/crops.csv',header=TRUE)

OptimalStation <- function(lat, long){

start\_date <- '2017-01-01'

end\_date <- '2018-01-01'

# Start of 1st function

lat\_lon\_df <- data.frame(id = "Station", latitude = lat, longitude = long)

#Load in all station metadata from previously saved file

options(noaakey = "YzLzNDLCXIzUwfAsWljYgxvxmZPMHtIj")

library(rnoaa)

load("/Users/Zach/Documents/Hydroinformatics 7460/AguaLibre/station\_data.Rdata")

#Retrieve the 10 closest weather station's metadata (distance in km)

closest\_stations <- meteo\_nearby\_stations(lat\_lon\_df = lat\_lon\_df,

station\_data = station\_data,

radius = 10)

closest\_stations <- as.data.frame(closest\_stations)

lat <- as.vector(closest\_stations$Station.latitude)

long <- as.vector(closest\_stations$Station.longitude)

#Retrieve data from closest weather stations (prcp in tenths of mm, temp in tenths of decC)

monitorIDs <- as.vector(closest\_stations[[1]])

climate\_data <- meteo\_pull\_monitors(monitorIDs,

var = c('PRCP'),

date\_min = start\_date,

date\_max = end\_date)

#Determine which stations have the most and least amount of available data

station\_data <- vector(mode = "list", length = nrow(closest\_stations))

for (i in 1:nrow(closest\_stations)){

df <- climate\_data[climate\_data$id == closest\_stations[i,1],]

id <- df$id[1]

avail\_prcp <- nrow(df)

if (avail\_prcp == 0){

avail\_prcp <- NA

id <- closest\_stations[i,1]

}

v <- c(id, avail\_prcp)

station\_data[[i]] <- v

}

station\_data <- do.call(rbind, station\_data)

colnames(station\_data) <- c('id', 'avail\_prcp')

#Populate the closest stations data with this new information

newcols <- c('AvailableData')

closest\_stations[, newcols] <- NA

for (j in 1:nrow(closest\_stations)){

closest\_stations$AvailableData[j] <- as.numeric(station\_data[j,2])

}

#Determine optimal station based on distance from user location and available data

closest\_stations <- closest\_stations[!is.na(closest\_stations$AvailableData),]

dist <- rank(closest\_stations$Station.distance)

dat <- rank(-closest\_stations$AvailableData)

tot <- dist + dat

rankings <- cbind(closest\_stations$Station.id, closest\_stations$Station.name, dist, dat, tot)

colnames(rankings) <- c('id', 'Name', 'Distance','AvailableData', 'Total')

row\_pos <- which.min(rankings[,5])

optimal\_station <- rankings[row\_pos,1]

optimal\_data <- climate\_data[climate\_data$id == optimal\_station,]

optimal\_data$SiteName <- id

station\_metadata <- closest\_stations[which(closest\_stations$Station.id == optimal\_station),]

prcp\_data <- optimal\_data

#Create map showing user location relative to nearby weather stations

library(ggmap)

bbox <- make\_bbox(long, lat, f=0.05)

map <- get\_map(bbox,maptype="toner-lite",source="stamen")

mapPoints <- ggmap(map) + ggtitle('Nearby Weather Stations: 10 km Radius, Optimal Weather Station:', station\_metadata$Station.name) +

geom\_point(aes(x = Station.longitude, y = Station.latitude, color = AvailableData, size = Station.distance),

data = closest\_stations) +

scale\_colour\_gradient(low = "purple", high = "cyan", na.value = 'purple') +

xlab("Longitude") + ylab("Latitude") + geom\_point(aes(x = Station.longitude, y = Station.latitude),

color = 'red',

shape = 18,

size = 3,

data = station\_metadata)

mapPoints

localMap <- mapPoints

ggsave("NearbyWeatherStations.png", width = 7, height = 5)

pathname <- '/Users/Zach/Documents/Hydroinformatics 7460/AguaLibre'

localMap <- paste0(pathname,'/NearbyWeatherStations.png')

# End of 1st function

# Start of 2nd function

# Load new packages

library(rjson)

library(jsonlite)

library(RCurl)

# Input

lat <- station\_metadata[3]

long <- station\_metadata[4]

start\_date <- '2000-01-01'

# Save base and full url as variables

url\_usu = "https://climate.usu.edu/API/api.php/v2/key=62PyzUjrCDYh0JB95faxrDcGB9tTss/evapotranspiration/average\_monthly\_sum"

full\_url = paste0(url\_usu,'/state=UT/network=ghcn','/lat=',lat,'/long=',long,'/start\_date=',start\_date,'/end\_date=',end\_date,'/units=m/month=(1,2,3,4,5,6,7,8,9,10,11,12)/buffer=10')

# Convert JSON to data frame

evap\_data <- fromJSON(getURL(full\_url))

# Break down list structure into numeric dataframe

evap\_data <- do.call(rbind, evap\_data[[2]])

evap\_data <- evap\_data[c(2:13),]

evap <- as.numeric(evap\_data[,1])

mon <- as.numeric(evap\_data[,2])

std <- as.numeric(evap\_data[,3])

evap\_data <- data.frame('month' = mon, 'evap\_sum' = evap, 'std\_dev' = std)

# End of 2nd function

output\_data <- list(prcp\_data, evap\_data, localMap)

#End

return(output\_data)

}

##ENDglobal##ENDglobal##ENDglobal##ENDglobal##ENDglobal##ENDglobal##ENDglobal##ENDglobal##ENDglobal##ENDglobal

######ui######ui######ui######ui######ui######ui######ui######ui######ui######ui######ui######ui######ui

######ui######ui######ui######ui######ui######ui######ui######ui######ui######ui######ui######ui######ui

######ui######ui######ui######ui######ui######ui######ui######ui######ui######ui######ui######ui######ui

ui <- fluidPage(

# Application title

titlePanel("AQUA LIBRE:

Rainwater Collection and Garden Irrigation Demand"),

sidebarLayout(

sidebarPanel(

numericInput("Lat","Enter Latitude of Site",

value = 40.767011, min = NA, max = NA, step = 0.000001, width = NULL)

),

sidebarPanel(

numericInput("Long","Enter Longitude of Site",

value = , -111.846033, min = NA, max = NA, step = 0.000001, width = NULL)

)

),

# Sidebar with a slider input for number of bins

sidebarPanel(

numericInput("Garden\_Area",

"Input Garden Area in Acres:",

value = 0.5, min = NA, max = NA, step = 0.001, width = NULL

)

),

sidebarPanel(

numericInput ("Roof\_Area",

"Input Roof Area in Acres:",

value = 0.8, min = NA, max = NA, step = 0.001, width = NULL

)

),

mainPanel(

#Text explaining how to find areas using the map

h6("Use the 'Create

New Measurement' tool in the upper right side of the map to trace

the outside of your garden. Input the area value into the 'Garden Area' box.

Repeat the process for measuring the roof area used for rainwater collection.

Input into 'Roof Area' box."),

leafletOutput("map1"),

imageOutput("map2"),

tableOutput("table1")

),

sidebarLayout(

sidebarPanel(

#making a selection box widget

selectInput("Chosen\_Crops", label = h4("Choose Your Crops"),

choices = unique(crops$crop),

selected = NULL,

multiple = TRUE,

selectize = TRUE,

width = NULL,

size = NULL),

# Making an Irrigation Efficiency Widget

sliderInput("Irrigation\_Eff", label = h4("Choose Your Irrigation Efficiency"), min = 0,

max = 100, value = 75),

h6("Common Efficiencies: Drip Irrigation 90%. Sprinklers 75%"),

# Making a Rooftop Efficiency Widget

sliderInput("Roof\_Eff", label = h4("Choose Your Rainwater Collection Efficiency"), min = 0,

max = 100, value = 75),

h6("Common Efficiencies: 90% for most roofs. New metal roofs up to 95%.")

),

mainPanel(

)

)

)

######ENDui######ENDui######ENDui######ENDui######ENDui######ENDui######ENDui######ENDui######ENDui######

######server######server######server######server######server######server######server######server######

######server######server######server######server######server######server######server######server######

######server######server######server######server######server######server######server######server######

server <- function(input, output, session) {

# Render interactive map for user to draw garden and roof areas

output$map1 <-renderLeaflet({

m<-leaflet() %>%

addTiles() %>%

addMarkers(lng = input$Long, lat = input$Lat, popup="Your Site") %>%

setView(lng = input$Long, lat = input$Lat, zoom = 16 ) %>%

addMeasure()

})

# Render map image showing user location relative to nearby weather stations

output$map2 <- renderImage({

data1 <- OptimalStation(lat = input$Lat, long = input$Long)

source <- data1[[3]]

list(src = source, contentType = 'image/png', width = 700, height = 600)

})

}

shinyApp(ui = ui, server = server)