CEEDS Weather Dashboard

Julia Lee, Marta Garcia, Mirella Hernandez

1 Department, Street, City, State, Zip

Abstract

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Curabitur eget porta erat. Morbi consectetur est vel gravida pretium. Suspendisse ut dui eu ante cursus gravida non sed sem. Nullam sapien tellus, commodo id velit id, eleifend volutpat quam. Phasellus mauris velit, dapibus finibus elementum vel, pulvinar non tellus. Nunc pellentesque pretium diam, quis maximus dolor faucibus id. Nunc convallis sodales ante, ut ullamcorper est egestas vitae. Nam sit amet enim ultrices, ultrices elit pulvinar, volutpat risus.

Author summary

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Curabitur eget porta erat. Morbi consectetur est vel gravida pretium. Suspendisse ut dui eu ante cursus gravida non sed sem. Nullam sapien tellus, commodo id velit id, eleifend volutpat quam. Phasellus mauris velit, dapibus finibus elementum vel, pulvinar non tellus. Nunc pellentesque pretium diam, quis maximus dolor faucibus id. Nunc convallis sodales ante, ut ullamcorper est egestas vitae. Nam sit amet enim ultrices, ultrices elit pulvinar, volutpat risus.

Text based on plos sample manuscript, see
http://journals.plos.org/ploscompbiol/s/latex

Introduction

Methods

Shiny app:

To solve our problem, we had many options on how to create a better way of displaying the weather data. We first discussed the possibility of making a new website in html but we ultimately decided to build a Shiny App using the Shiny and Shiny Dashboard packages in R. Shiny is a package that allows us "to build interactive web applications (apps) straight from R." Shiny allows the user to be able better communicate data with interactive charts, visualizations, text and tables. We then decided to use Shiny Dashboard which is a package that allows us to create an interactive dashboard using Shiny. Because our problem was to find a way to display interactive data visualizations, the Shiny package seemed to be the better option because that is what it was built to do. Shiny also allows us to make normally static plots like ggplots interactive and take in user input so we don't need to use html widgets.

Shiny is based on a reactive programming model which means that it takes in input given by the user of the app and acts accordingly. Shiny apps have two components: the user interface (UI) and the server. You can think of the UI as what the user of your app will see when they open your app. The UI component converts your code in R and generates it into a web document in html. UI contains the instruction about what you want the user to see and the layout of your app. our UI is where we define the dashboard page. And inside this dashboard page we define our elements like the sidebar menu, tabs, the dashboard body (Tab items, boxes, text). The server is the R code that tells your shiny server what to do when the user does certain things in the app. The server is a function where we define the output and tell the host of our app what to do with the user input. Server function is where we render the charts, visualizations, and tables. We then knit together the Server function and UI using the ShinyApp (UI, server) function in the Shiny package.

So we built a Shiny dashboard. Our dashboard has a sidebar menu that has four tab items. One tab item we built was the current weather tab. The current weather tab gives the user the the current 10 minute weather for WhatelyMet and OrchardMet. Because we wanted to have our data be shown as a text object and not a table, our code for this tab was largely done using html. another tab on our dashboard is the about tab where we explain our project and give information about the weather stations. We used a bit of HTML to change the sizing of the text and the alignment of the rows. To make the teal boxes, we used the shinydashboard Plus package, allowed us to make a clear, concise first page of our Shiny app.

Another tab that we built was the historic data tab. We made a way for the user to switch between daily data for OrchardMet and WhatelyMet in the historic data tab, this means the graphs shown are reactive and are based on the data set that was selected by the user. We did by making a reactive function inside the server that allows people to switch between data sets, this function we call datasetInput. We made graphs mapping 3 (Wind speed, precipitation, temperature) variables over time. To do build these graphs we used the Highcharter package that allows the user to filter the data based on a certain time period and high charter also has a feature that enables users to download the data used to make the graphic.

We also made a visualization where the user can make a custom correlation graph by picking the variables that they want. We made this correlation scatterplot by using ggplot and ggplotly to make the graphic more interactive. We also made a windrose graph to show wind speed and direction using ggplot. A wind rose is a polar chart which shows the user how wind speed and wind direction are distributed at a over a specific period of time. We also created a way to allow the user to choose a variable and get summary statistics on that variable. In this tab, we made the graphs be in a fluid page, this allows for the size of the boxes and visuals to change based on the size of the window. We then put all of our content for the tab in separate boxes that are collapsible. We did this to minimize users' scrolling. The last tab we made is the raw data tab. This tab allows the user to filter the data based on columns and the user can also choose between daily data for OrchardMet and WhatelyMet. This tab also allows the user to search the data table using the DT package.

Ceeds package:

The data from the two weather stations is currently being saved on the Macleish Server hosted at Smith college. We used the Macleish package to fetch the two data tables from the server. However this involved a lot of code that would have to be repeated anytime we wanted to run the app. So we decided to write an R package to more efficiently do these tasks. This package was also built Help other students to work on the MacLeish weather data. We wrote a function that uses the Macleish package and

fetches 2 data sets (Whately, Orchard) that is updated every ten minutes. We also created a function,get_daily(), inside the package that takes a data set and gives the user the data grouped by date. Our data from the server comes in 10 minute increments but we might want to group by date. So we used the Lubridate package was very helpful. It allowed us to Parsed timestamps into dates (turning variables into dates that r can recognize as such). This allowed us to group our data by date.

References