CEEDS Weather Dashboard

Julia Lee, Marta García, Mirella Hernandez

Abstract

We were tasked with finding a way to better showcase weather data from the MacLeish Field station, as well with incorporating interactive graphics and allowing users to download and filter the data. Due to the capabilities of Shiny, we decided to use the shiny and Shiny Dashboard packages in R to build an interactive dashboard. We also wrote a package to get the data from the MacLeish server.

Introduction

The Center for the Environment, Ecological Design, and Sustainability (CEEDS) is located in Wright Hall building of Smith College. The goal of Ceeds is to train and allow students to consider sustainability issues across various disciplinary areas to integrate that multi-disciplinary knowledge in support of environmental decisions and action. Macleish Field Station is one of the many opportunities Ceeds provides to the Smith community to pursue environmental research, outdoor recreation, and low impact recreation.

The MacLeish Field Station is located in a 260 acre of forest farmland near Whately, Massachusetts, and its weather collection sites are at the end of Poplar Hill Road. There are two weather site locations in the field, one called WhatelyMet Tower and the other one called OrchardMet. The WhatelyMet tower is mounted at the top of a 25.3m tall tower, and it is 250.8m above sea level. It was initially created to collect air pollution data. In order to collect more efficient weather data, OrchardMet weather station was created. The OrchardMet weather station is 10m tall located in a forest clean area close to ground level. However, the trees do not allow OrchardMet weather station to collect data efficiently as hoped.

As part of the SDS 410 Capstone class, our group of 3 were given Macleish weather data from the Macleish package that is collected through the Smith server. There is a current Macleish weather dashboard that collects a variety of weather data: date, wind speed, temperature, wind direction, real humidity, pressure, solar radiation, and rainfall. The server has been collecting data since January 2012, and it is updated every 10 minute increments. Thus, there is over 377,016 entries. The data loggers run on solar power and stores the data so even if the server is down the data is saved. The current MacLeish weather dashboard[1] contains a lot of useful information, and the data is organized into readable tabs. Because the data is trapped in individual "data silos", it is not readily accessible. Therefore, the goal for the given project is to recreate the current Macleish weather dashboard to illuminate the data and make it more user friendly, it download accessible, and it visually interactive as proposed by our client, Paul Wetzel. Our client Paul Wetzel is the field station manager of MacLeish Field Station, and he is the main user of the current website. As discussed and mentioned

[1] current Macleish weather dashboard: http://macleish.smith.edu/index.html

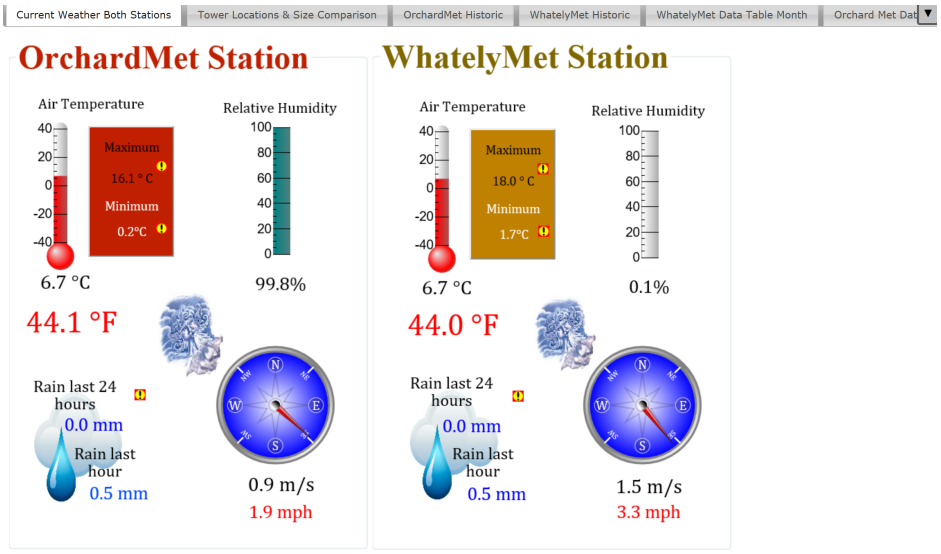

Fig 1. Example of a what the current Macleish weather dashboard looks like

earlier about our clients needs, we chose to make the new weather dashboard using the Shiny package. We choose to use Shiny, since it would allow us to make the download data opportunity catered to the audience's needs, and it would help us make the graphs interactive and user friendly. We believe our focus would supply to the current target audiences, Paul Wetzel, other Smith students, Smith faculty/researchers, and Northampton Department of Public Works.

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 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

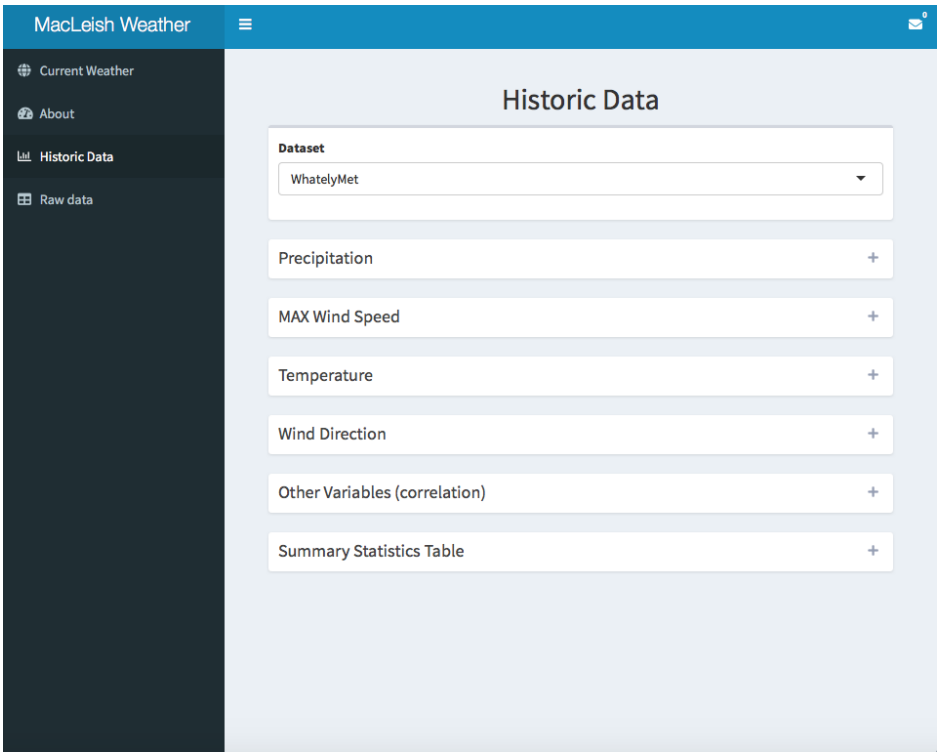

Fig 2. Example of a what our sidebar menu and dashboard looks like

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. 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 on our dashboard is the about tab where we explain our project and give information about the weather stations.

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

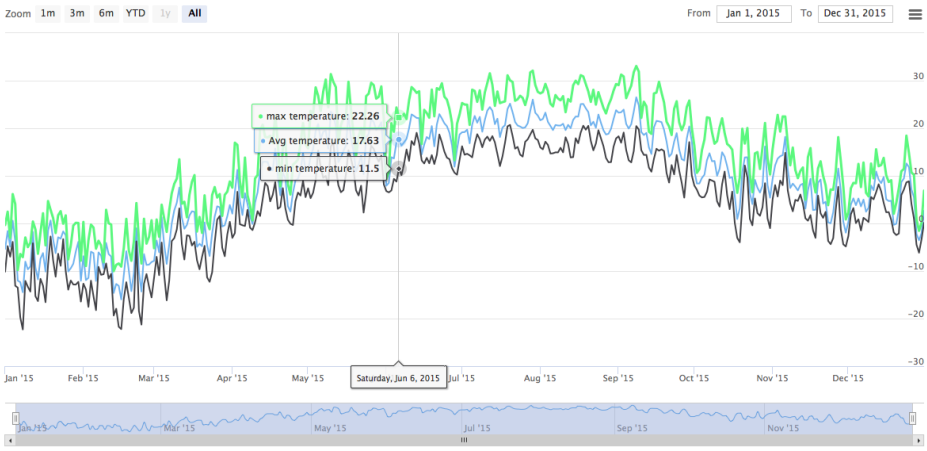

Fig 3. Example of a Highcharter graph that shows temperature

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 called "ceeds" 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 that 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.

Issues

Throughout this whole process, our group has faced a variety of issues and consequently, temporary limitations. At the beginning of our process, we first had to learn to use the Shiny package. There was a learning curve in learning to build a web app and we initially struggled with building the first parts of the app and figuring out how to integrate the R knowledge we had with this new package. We took the DataCamp course to learn the basics of Shiny and from there, dove straight into our new data, once we had access to the data. Since we were reliant on the MacLeish package, which relied

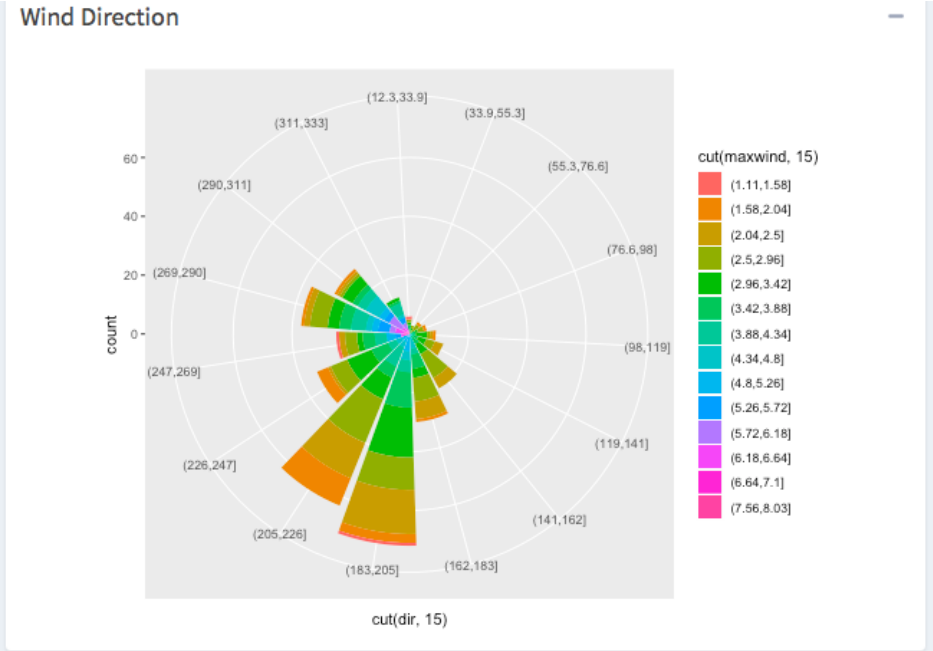

Fig 4. A wind rose graph

on the MacLeish server to receive the data, whenever the MacLeish server was down, we struggled with working with the data we needed to move forth with the project.

Also at the start of this project, we met with our client to gather more information of what he wanted and what we felt we could deliver knowing our skill set in R and Shiny. Since Wetzel was not familiar with the capabilities of R, it was up to us to fill him in with what we thought was possible. These plans included reducing the amount of tabs, changing all the graphs to being clearly interactive, and making the web app overall more user-friendly and comprehensive. We did not create a printed mockup until much later in the process so it was difficult to communicate our vision to him and for him to understand the potential of R and Shiny. However, in our biweekly meetings with him, we filled him in with whatever progress we were making and once we started creating visualizations using highcharter, he was able to better understand the kind of graphs we could create and integrate into the new web app.

In the similar vein as to how Wetzel did not know much about R, our group did not know much about weather data and the importance of all the variables. During our meeting Wetzel taught us the importance of certain variables but also told us that we should focus on the precipitation, temperature and wind variables if we needed to narrow our scope. In teaching us about the other variables, such as soil temperature and server temperature and this helped us figure out which variables were most important to the user. For example, the server room temperature would be most useful to the weather station manager (Wetzel) because that temperature would allow him to see when the room was overheating or too cold, and not very functional for a user looking to download weather data for a school project. Through Wetzel, we also learned that temperature data is only accurate up to the thousandth place which made us modify our app to reflect that accuracy.

Arguably our largest and most time-consuming issue, making tabs work in Shiny took us a while. Since our project is so visual, and tabs played a major part in our functioning dashboard, this set us back and we were unable to move forward with the HTML & CSS coding until we could solve this problem. With some help from our

professor Ben Baumer, we were able to get tabs to work. Through this, we learned the importance of correct placement of commas and parentheses in Shiny since as we discovered later, that had been our problem the whole time. Once tabs worked, we were able to put all the appropriate graphs in each tab and start working on the user interface that we coded mostly with HTML & CSS.

A smaller issue that we had was making graphs reactive. Since we were so familiar with using ggplot package through tidyverse, we wanted to have all our data visualizations be programmed using ggplot. However, we had a lot of issues with making these visualizations reactive with Shiny, which posed a problem of how a user would then interact with these graphs. After trying for a short while, we ultimately moved onto a different package called highcharter, which made the graphs interactive and also allowed an element of downloading data that we expanded on later with a tab we included.

Limitations & Moving Forward

There are many things that can looked at in future to expand this project. A limitation of our project is that we have not yet surveyed our target audience about whether our app is more usable than the current dashboard. Audience feedback would help us moving forward in reach our end goal of creating a user friendly weather dashboard. Right now our dashboard has very few features and could have more places for user interaction. For example, Our wind rose graphic is not interactive which could improve this project. Another feature we might want to have in the future isa way to allow users to identify and filter out outliers. After talking to our client, we found out that sometimes the weather equipment breaks and leads to wrong data but we don't necessarily want to filter out possible outliers of the data assuming that they mean equipment malfunctions so we might want the users of dashboard to make that decision to not mislead them. We might also want to have more variables in our data set like soil temperature which is used to study climate change.

References

Our Github Repository: https://github.com/beanumber/ceeds