CEEDS Weather Dashboard

Julia Lee, Marta García, Mirella Hernández

Abstract

The MacLeish Field Station is open to all the Smith College community, and the field station contains and captures a lot of weather data and historical information, such as precipitation, temperature, and wind speed. Thus, the station currently has a weather dashboard where all the information and data from the year 2012 of the field is collected and displayed. Despite there being an existing MacLeish weather dashboard, the MacLeish Field Station manager, Paul Wetzel, wants to update the weather dashboard to better showcase the weather data for the public to feel more inclined to download the MacLeish weather data and overall use the dashboard. Therefore, we primarily focused on creating the updated MacLeish weather dashboard to be more user friendly by incorporating interactive graphics and allowing users to download and filter the data. We created a Shiny App to update the dashboard, so we used "shiny" and "shiny dashboard" packages in R as well as HTML and CSS to build and interactive dashboard.

Introduction

The Center for the Environment, Ecological Design, and Sustainability (CEEDS) is located in Wright Hall building of Smith College. The Macleish Field Station is a part of CEEDS, and students are able to conduct research, work on environmental projects, and overall explore the field there. The MacLeish Field Station is located near Whately, Massachusetts, and its weather collection sites are at the end of Poplar Hill Road (footnote/cite current MacLeish weather dashboard). There are two weather site locations in the field, one called WhatelyMet tower and the other tower is called OrchardMet. The WhatelyMet tower is 25.3 meters tall and it is slightly further away from the center of the field station, since it was initially created to collect air pollution data. In order to collect more efficient weather data on ground level, OrchardMet weather station was created. Despite this, the trees do not allow OrchardMet weather station to collect data efficiently as hoped.

As part of the SDS 410 Capstone class, we worked with Paul Wetzel, the MacLeish field station manager, in updating the current MacLeish weather dashboard. The current MacLeish weather dashboard 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 through WhatelyMet tower, and it has been collecting additional weather data since 2015 through OrchardMet tower. The data is updated every 10 minutes; 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 (see fig. 1). Our client, Wetzel, is currently the

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

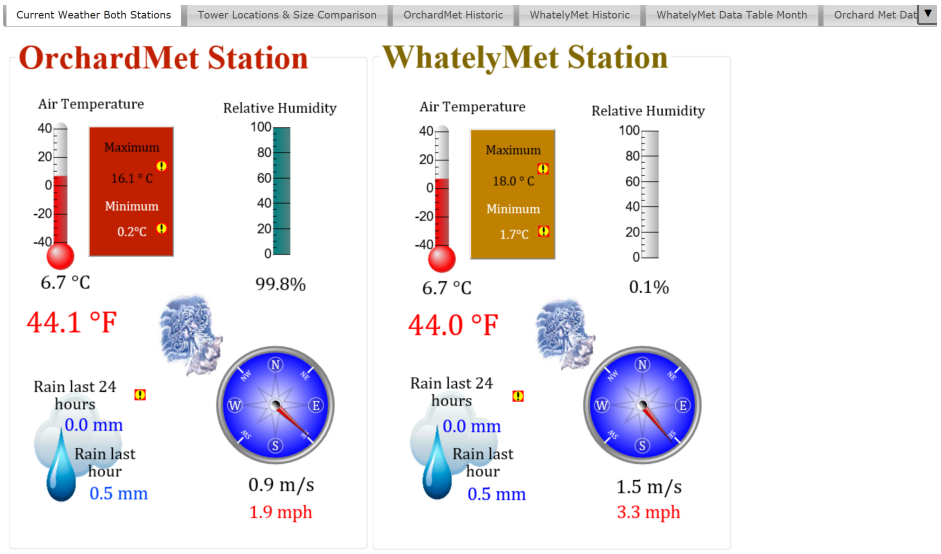

Fig 1. Example of a what the current Macleish weather dashboard looks like

one who uses it the most and understands it. Despite this, Wetzel wants to update the MacLeish weather dashboard for others in addition to himself to better access and use the MacLeish weather data and its dashboard. Therefore, the focus we chose for this project is to recreate the current Macleish weather dashboard to better showcase the data and make it more user friendly by keeping its tab structure, adding visually interactive graphs, and allowing users to download and filter data as proposed by Wetzel. We chose to create a Shiny App [1] when updating the MacLeish weather dashboard, so we used "shiny" and "shiny dashboard" packages. We choose to use Shiny since it would allow us to make the download data opportunity catered to the audience's needs, it would help us make the graphs interactive, and it be more 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 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, an informative platform that visually collects, analyzes, and displays data, 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

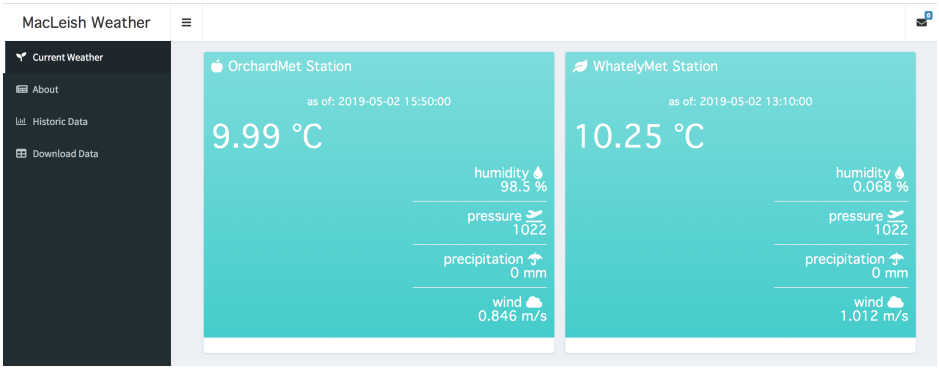

Fig 2. Current Weather Tab displaying WhatelyMet and OrchardMet current weather and showing how our sidebar menu and dashboard looks like

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. Inside the 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. The server function is also 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.

Current Weather Tab:

Thus, we built a Shiny dashboard. Our dashboard has a sidebar menu that has four tab items: current weather, about, historical data, and download data. We chose to focus on these 4 tab items since it answers to our client's primary needs. The first tab we built and have displayed as the main page of the weather dashboard is the current weather tab (see fig. 2). The current weather tab gives the user 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. Additionally to address our clients need of having the MacLeish weather dashboard be more user friendly, we chose to design the weather tab in a minimalist style. By doing so, the users will be able to better see the data in one glimpse.

About Tab:

The second tab we built on our dashboard is the about tab, which is where we explained our project and have information about the weather stations (see fig 3). We also added links for the public to contact Wetzel for further questions and interests as well as our repository link [2]. This tab was created as the updated version of the "Tower Locations & Size Comparisons" tab in the current MacLeish Dashboard to better communicate with the users. 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 [3], which allowed us to make a clear, concise first page of our Shiny app.

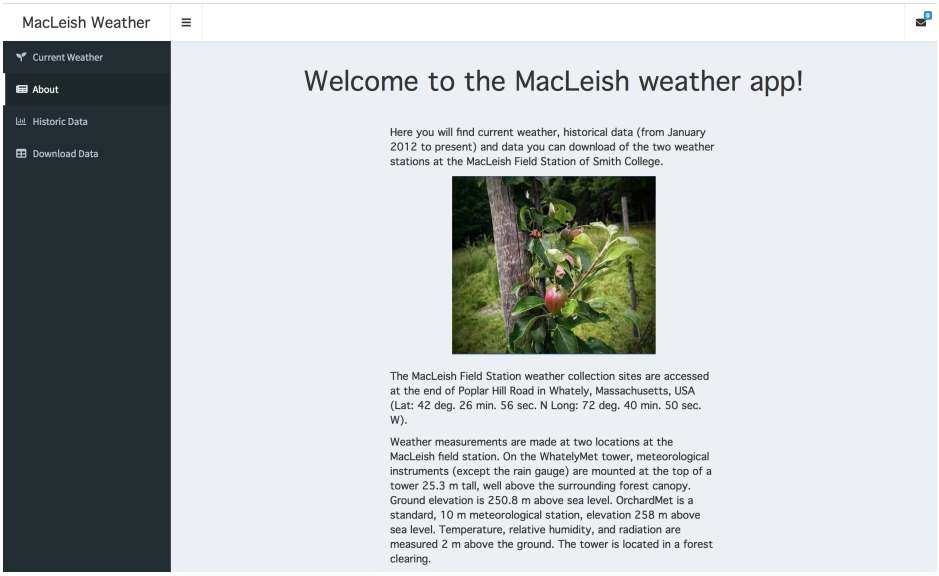

Fig 3. Welcoming the users and letting them know who to reach out to for further information and how the app works

Historic Data Tab:

The third tab we built was the historic data tab (see fig 4). This tab is the updated version of the "OrchardMet Historic" and "WhatelyMet Historic" of the current MacLeish weather dashboard. When building this tab, we focused on displaying the weather data through visual interactive graphs as our client requested. 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 this by making a reactive function inside the server that allows people to switch between data sets. We made graphs mapping 3 of our client's focus variables (Wind speed, precipitation, temperature) over time (see fig 5). To build these graphs, we used the Highcharter package [4] 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 (see fig.6). We also made a windrose graph to show wind speed and direction using ggplot (see fig 7). 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 (see fig 8). 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.

Dowloand Data Tab

Since our client wanted users to be able to download and filter the MacLeish weather data that is displayed in the dashboard, we added a new tab, raw data tab, which the current MacLeish weather dashboard does not have. This tab allows the user to filter the data based on columns and the user can also choose between daily data for

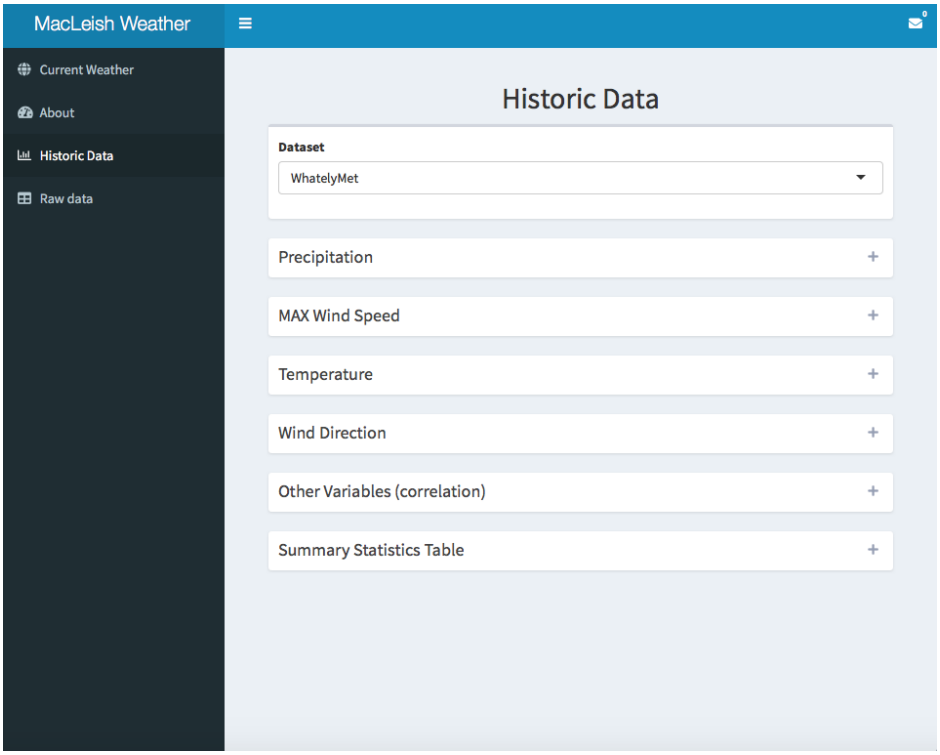

Fig 4. fig 4: The way the Historic Data tab looks and how it allows the users to choose what weather station they would like to look at

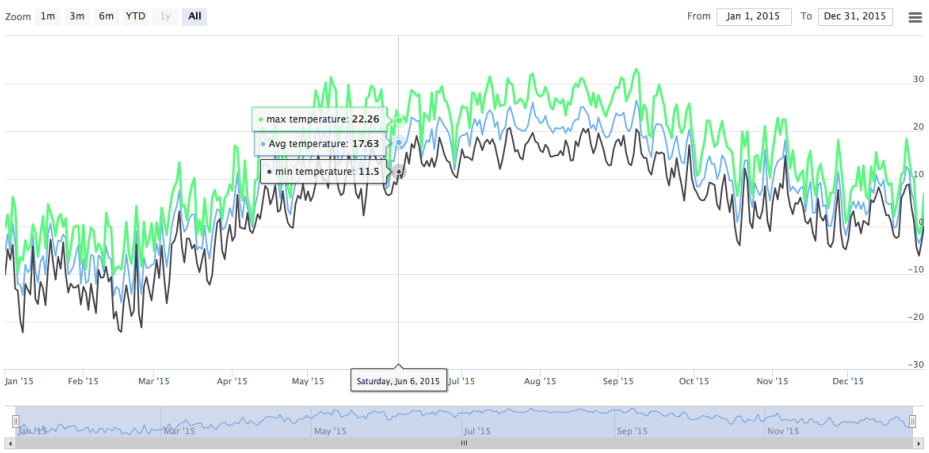

Fig 5. Example of a Highcharter graph that shows temperature

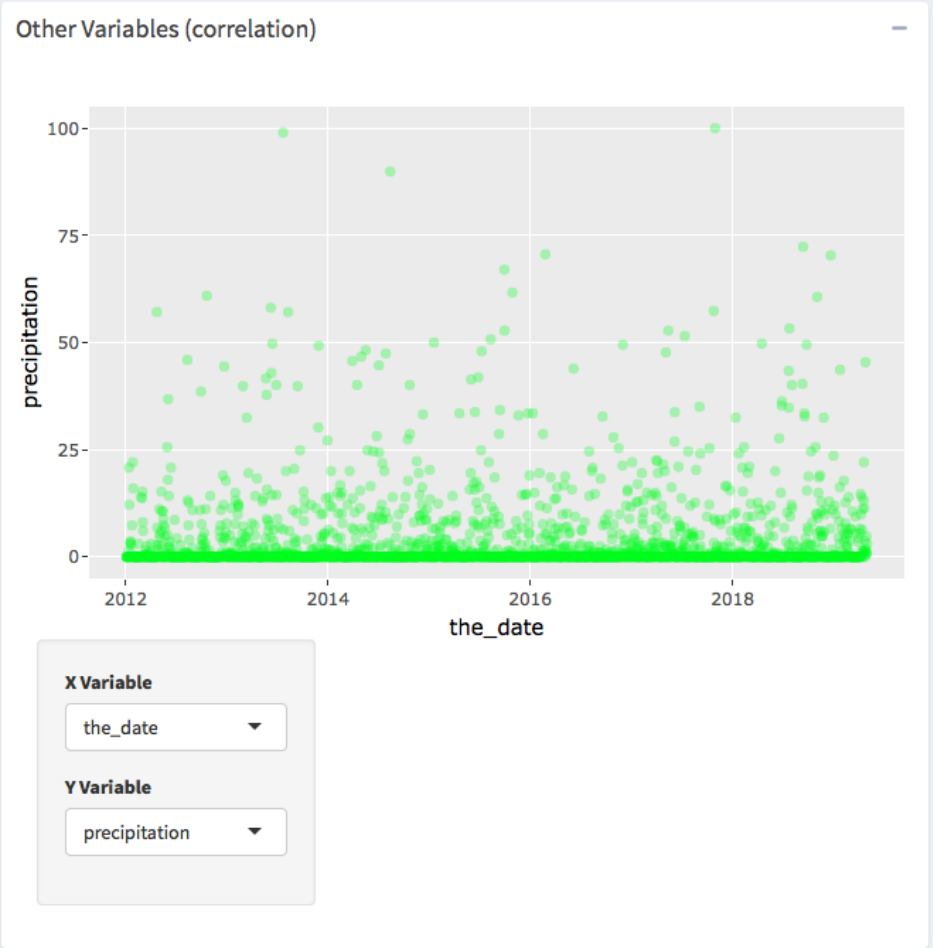

Fig 6. a correlation scatterplot

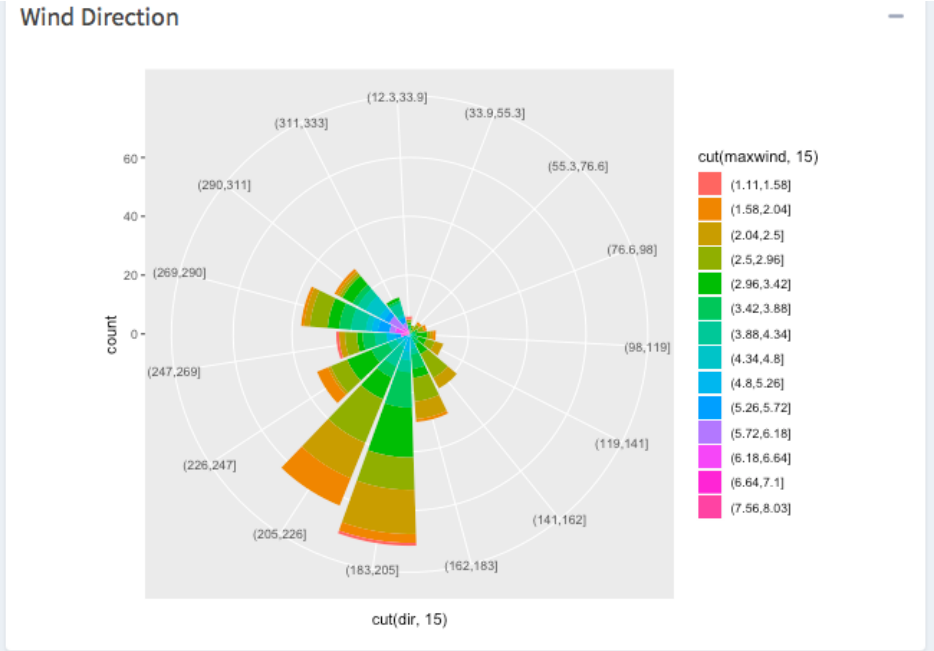

Fig 7. A wind rose graph

OrchardMet and WhatelyMet (see fig 9). 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 [5] 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 to 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 [6] was very helpful. It allowed us to parse timestamps into dates (turning variables into dates that R can recognize as such). This allowed us to group our data by date. [7]

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 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.

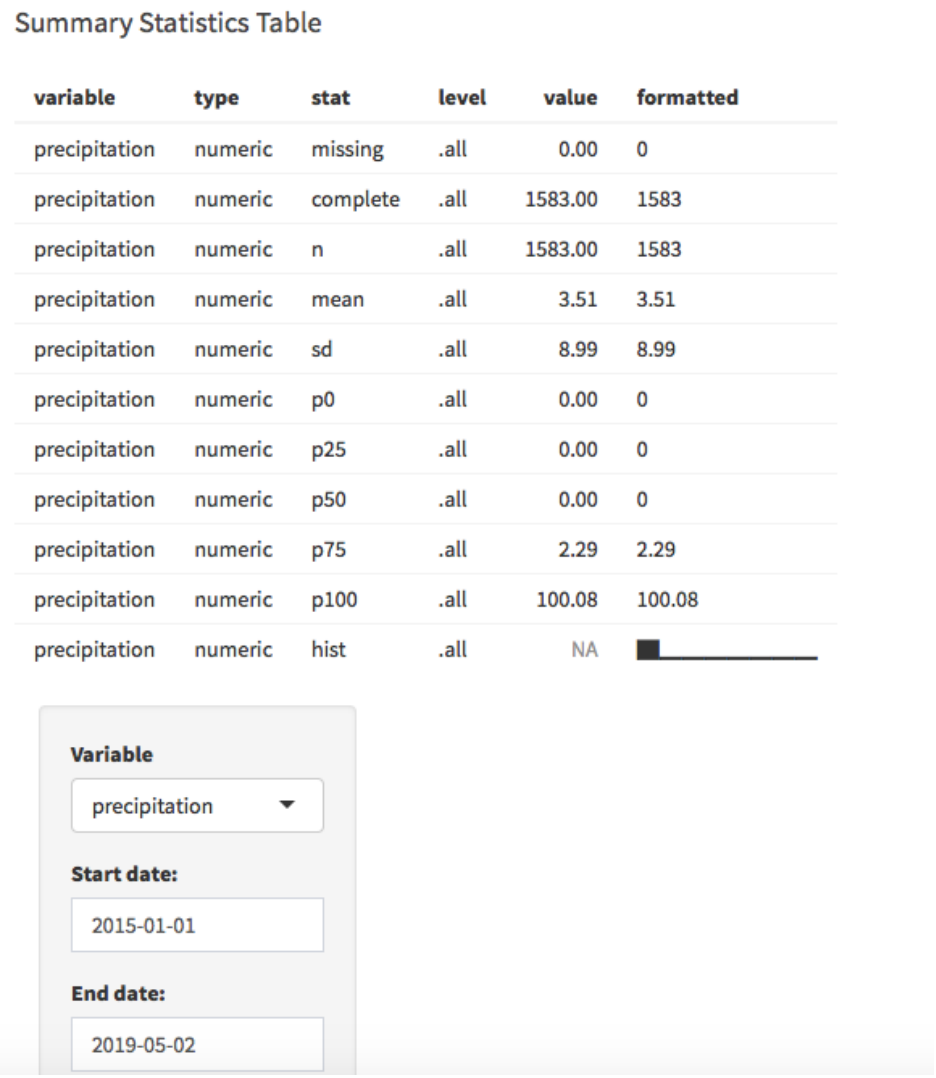

Fig 8. statistics summary table

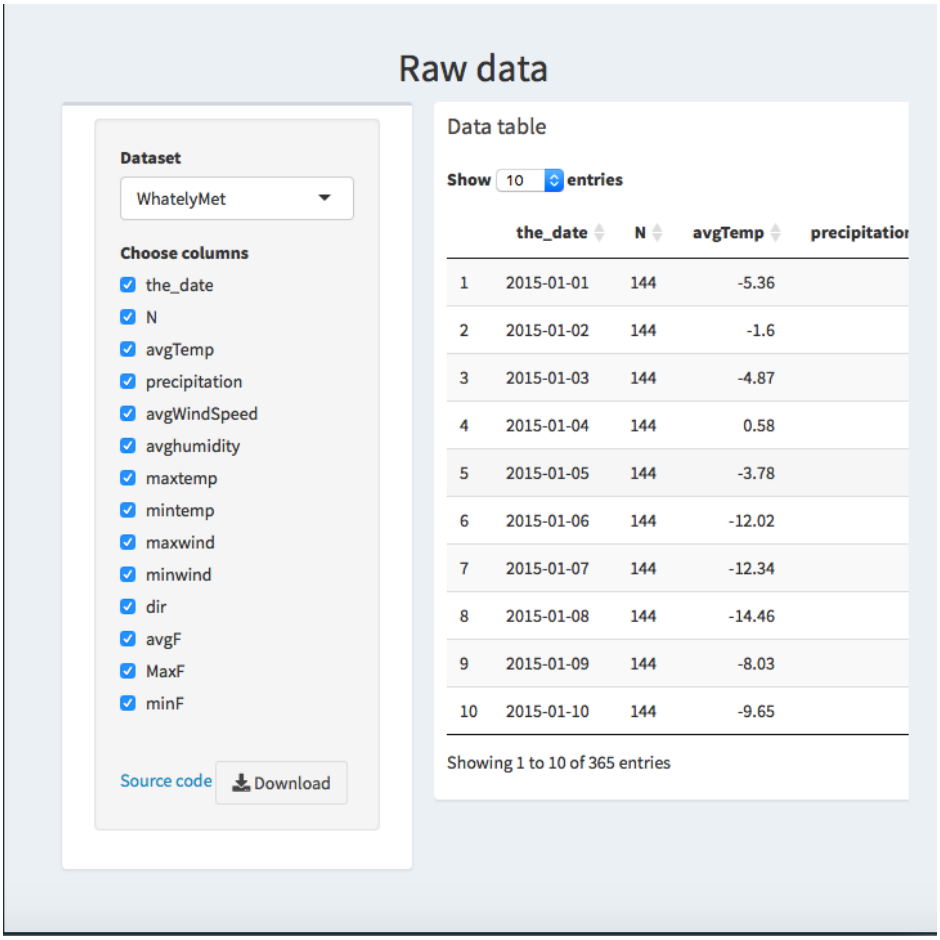

Fig 9. The way dowloading data tab looks, and how it allows users to filter the data they would like to download

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 to figure out. 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. We had already programmed in some graphs but with tabs not working, they would all appear on the same page and when the user tried to click to different tabs, they would not switch. However, 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. We also faced issues when building the tab that allowed users to download data and making images show up in the about tab. Both of these issues were solved by continuously trying new methods over and over again, but specifically solved the issue with images by installing the package base64enc since the problem was about different directories and file paths that were not accessible by all machines.

Moving Forward

There are many things that can looked at in future to expand this project. A future endeavor 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. We have already tested our dashboard with a group of peers, however we would like to test on a wider audience.

After talking to our client, we found out that sometimes the weather equipment breaks which leads to inaccuracy in the data it records. Due to this, Wetzel often goes in and manually changes the data. We don't necessarily want to filter out possible outliers of the data (assuming that they mean equipment malfunctions), but we would have liked to either include a small disclaimer that would tell people downloading or analyzing the data about this or include both the raw and corrected data. That way, users can reach that conclusion on their own just by comparing the two datasets.

We would have also liked to implement some sort of special setting that would have extra tabs for our client, the weather manager. These extra tabs would include data that is important to him, such as soil temperature and the bunker room temperature, both tabs we felt were not important to our general audience of students and others looking to check the weather or download a large amount of data. A variable like soil temperature is used to study climate change so while it may not seem important immediately, with context, it could prove to be a valuable variable to study and analyze.

References

1. Chang W, Cheng J, Allaire J, Xie Y, McPherson J. Shiny: Web application framework for r [Internet]. 2018. Available: `https://CRAN.R-project.org/package=shiny`

2. Baumer BS, García M, Hernandez M, Lee J. Ceeds github repository. `https://github.com/beanumber/ceeds`; 2019.

3. Granjon D. ShinydashboardPlus: Add more 'adminlte2' components to 'shinydashboard' [Internet]. 2019. Available: `https://CRAN.R-project.org/package=shinydashboardPlus`

4. Kunst J. Highcharter: A wrapper for the 'highcharts' library [Internet]. 2019. Available: `https://CRAN.R-project.org/package=highcharter`

5. Baumer BS, Goueth R, Li W, Zhang W, Horton N. Macleish: Retrieve data from macleish field station [Internet]. 2018. Available: `https://CRAN.R-project.org/package=macleish`

6. Wickham H. Tidyverse: Easily install and load the 'tidyverse' [Internet]. 2017. Available: `https://CRAN.R-project.org/package=tidyverse`

7. Baumer BS, García M, Hernandez M, Lee J. Ceeds. 2019.