STA242 Final Project

Weather Data Interactive Visualization

XueTing Shao, Linfeng Li, Kefu Yu

Tuesday, June 10, 2015

git@bitbucket.org:linfenglidy/sta242project.git

Introduction

Weather visualization is always an interesting topic. We build our own American Weather Map, combined with water usage, which is inspired by California's water shortage.

In the construction, multiple sources of data is cleaned, combined and matched to visualize water usage per person on the USA map. Shiny package is applied to make the map interactive and flexible. And temperature prediction is done by ARIMA model.

Data Process

Data Source

We planned to find an API access since we are uncertain if it is legal to scrap data directly from website. However, it turns out that most APIs are not free and we can only request data limited times in five minutes through free APIs. Additionally, free and unlimited API is very slow. It is not suitable to use in an instant information display. Fortunately, there is an R package called weather Data we can use to fetch data. It provides several functions to read weather data from URL's(www.wunderground.com) and convert them as a clean R data frame.

But the challenge is that the data is based on airport/station identification instead of city/state, which means it is hard to locate the data to each state/county/city in the interactive map. Our solution is trying to match each airport in the weather data with the city name of that airport in the airport information data (Source: http://ourairports.com/data/). Then the weather data in a city can be approximated by those observed in the city's airport.

However, initial data are in different format and may contain unnecessary variables. Plenty time is spent on data cleaning and matching.

Data Profile and Cleaning

The raw airport data includes the airport identification from all over the world, and some airport has no longer in service, so our next step is to detect and removing the airport has been shut down and subsetting airport in the United States in order to improve the quality of our data. As a result, 444 airport identifications were chosen across the United States. One row of airport data is shown in Table 1.

Table 1 Airport Data (One Row)

Airport Id	**Name**	**Latitude**	**Longitude**
PHNL	Honolulu International Airport	21.3187	-157.922
Country	**Region**	**City**	**Scheduled_Service**
US	US-HI	Honolulu	yes

Then we try to match the airport data with weather data, using airport id as key. The weather data is obtained by function getSummarizedWeather() in weatherData package. Each observation in weather data has 24 variables in a day and airport. One row of the weather data of airport ID: PHNL is shown in Table 2. The weather data and airport location is well matched so they can be used directly.

Table 2 Weather Data (One Row)

Date	Max_TemperatureF	Mean_TemperatureF
2013-08-10	85	79
Min_TemperatureF	**Mean_VisibilityMiles**	**Mean_Wind_SpeedMPH**
73	10	10
CloudCover	**Events**	**WindDirDegrees**
6	Rain	48

However, the geometrical information in the water data has different format. Since we want locate the each county's water usage per person on the map, the county name and order should match with those in R's map() function. Counties data shown in Table 4 has the same order in county with those in map() function.

Table 3 Water Data (First Five Rows)

	STATE	COUNTY	Water Usage
1	AL	Autauga County	66
2	AL	Baldwin County	78
3	AL	Barbour County	74
4	AL	Bibb County	79
5	AL	Blount County	67

Table 4 Counties Data (First Five Rows)

	Name	Population	white	black	hispanic	asian
1	alabama,autauga	54571	77.2	19.3	2.4	0.9
2	alabama,baldwin	182265	83.5	10.9	4.4	0.7
3	alabama,barbour	27457	46.8	47.8	5.1	0.4
4	alabama,bibb	22915	75.0	22.9	1.8	0.1
5	alabama,blount	57322	88.9	2.5	8.1	0.2

As we can see, state name in water data is abbreviation and county name is following by county/city/parish. While, the location name in Counties data is a combined string with full name of state and name of county. To match them, we need another reference table called USAState (Source: http://www.50states.com/abbreviations.htm#.VSsXHFxWKfQ).

Table 5 USA State (First Five Rows)

	US State:	Abbreviation:
1	Alabama	AL
2	Alaska	AK

3	Arizona	AZ
4	Arkansas	AR
5	California	CA

After combine the three data tables, we get a cleanwater data frame which has the same order as the counties in function map().Although there are no water usage per person in some counties, the number and area of them is small. We just ignore them when we use water usage data to color the map.

Table 6 Clean Water

	STATE	COUNTY	DO.PSPCp
1	AL	Autauga County	66
2	AL	Baldwin County	78
3	AL	Barbour County	74
4	AL	Bibb County	79
5	AL	Blount County	67

Visualization

American Map

To make the visualization interesting, we decide to use a map to locate the place instead of a select box. Additionally, we are interested in the relationship between water usage and local weather conditions. Thus the map is colored based on the water usage per person in each county.

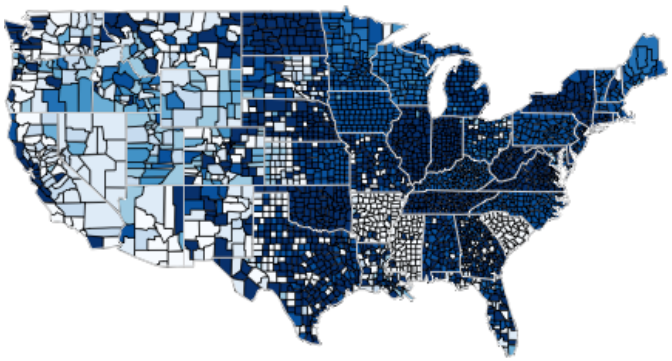

Figure 1 American Map(Water Usage Per Person)

As the map shows, water usage per person seems to be generally higher in east than west from color darkness. This may relate to the fact west coast encourages people to use water efficiently in terms of water shortage. In the middle part of America, the advanced agriculture and lower population may lead to higher water usage per person.

Interactive Plot using Shiny

It is straightforward to show the temperature information all over the country by using an interactive plot. A great way is using Java script, but we never know Java before. And R is a better tool to process data, so we decided to using Shiny package to build the interactive plot.

For the interactive plot, we hope we can achieve following goals:

(1) **Mouse interaction:** it is fun to click one point on the map and get the information we want.

To be able to use the data from the clicked point, functions reactiveValues(), nearPoints() and observeEvent are applied in the server script. This process is done by matching the axis values of the point in a table.

In our case, the number of weather observation stations is limited and not evenly located in each city. It is impossible to use every clicked point as station/airport ID to request weather data. Therefore, the nearest airport ID is used to represent the clicked point. We spread all airports on the map, as red circles shown below. (All red circles are set in transparent color in the final interface)

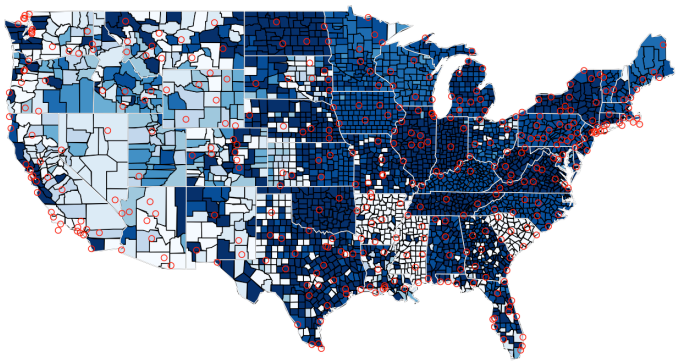

Figure 2 American Map(Airports)

Once we click a point on the map, nearPoints will get corresponding axis values (longitude and latitude) of the nearest red circle, and match the axis values in the reference table (airport.rda) to get the city name and airport ID.

(2) **Instant local weather report:** for each point clicked, we want to get the current local weather data for that location.

As mentioned before, current local weather data is obtained through getSummarizedWeather() in weatherData package, passed to the reactive values, transformed to HTML and displayed through renderUI() function in the user interface.

After clicking the point on the map, function checkSummarizedDataAvailability() will automatically check the data

availability, or shiny will force quit the app when the data is not available. Then the app will generate data.

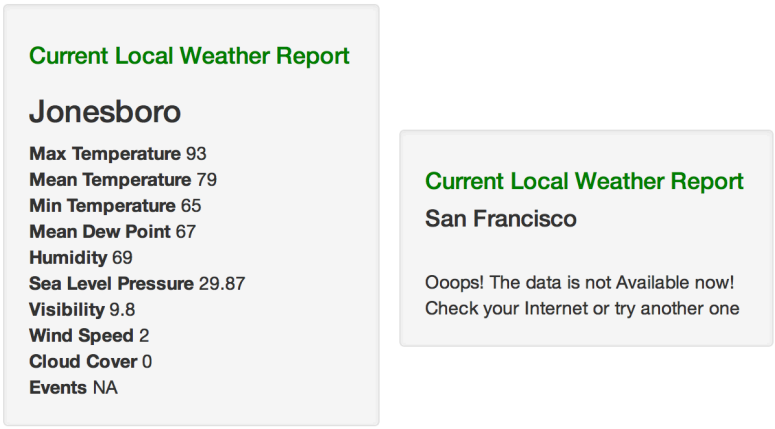

Figure 2 Weather Report (Two Cases)

(3) **Historical data trend:** for each point clicked, it is meaningful to get the historical weather data trend for that location.

The historical weather data is obtained by getSummarizedWeather() in weatherData package by setting a certain time period. In default, it starts from the date one year before to current date.

Meanwhile, the ARIMA model discussed later use historical data to predict next day's temperature.

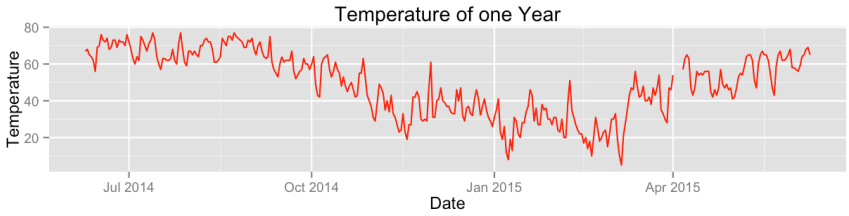

Figure 3 Historical Data Plot

(4) **Data flexibility:** it is useful to make comparison by allowing user to change time period and type of data showing in the plot.

The time duration and type of data can be changed through the control widgets in the user interface.

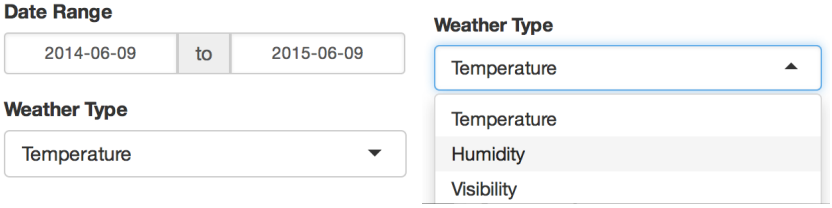

Figure 4 Control Widgets

Prediction

Time series analysis and forecasting has become a major tool in many applications to study trends and variations in variables like rainfall, humidity, temperature, and winds. The objectives of this section are to build ARIMA models based on the one year historical weather data that we fetch from the data source website with the airport ID across the United States. We use the build-in Auto-ARIMA() function in R to select our best appropriate order of the ARIMA model for each type of weather conditions. Additionally, we will carry out short-term prediction for the weather conditions.

ARIMA Model

ARIMA model stands for autoregressive integrated moving average and they are sometimes called Box-Jenkins models. In the ARIMA model, we have to choose an autoregressive model of order p is classified as AR (p) and a moving average model with q term is known as MA (q). A combined model that contains p AR-terms and q MA-terms is called an ARMA (p, q) model. Since our weather data is non-stationary, we also want to compute the time-shifted lags (d) for the non-stationary time-series data.

Since many climates usually follows a seasonal and annual cycle, it is more appropriate to use a seasonal ARIMA (p, d, q) model, whereby p is the order of the seasonal AR-model; d is the order of seasonal differencing and q is the order of the MA-model, the below tables shows the ARIMA models obtained for the maximum, minimum and mean temperature, and also the mean humidity time series at Lebanon, VA with airport id 04VA, along with their corresponding AIC-values .

Table 7 AIC Values for Each Variable

Variables	Station id	ARIMA MODELS	AIC
Max temperature	04VA	(1,1,3)	2500.94
Min temperature	04VA	(0,1,3)	2421.987
Mean temperature	04VA	(0,1,3)	2329.983
Mean Humidity	04VA	(2,1,2)	2741.157

In the next step, we are going to verify if the identified ARIMA model is appropriate. We need to look at the ACF and PACF of the selected models residuals. Those are shown in the figures below.

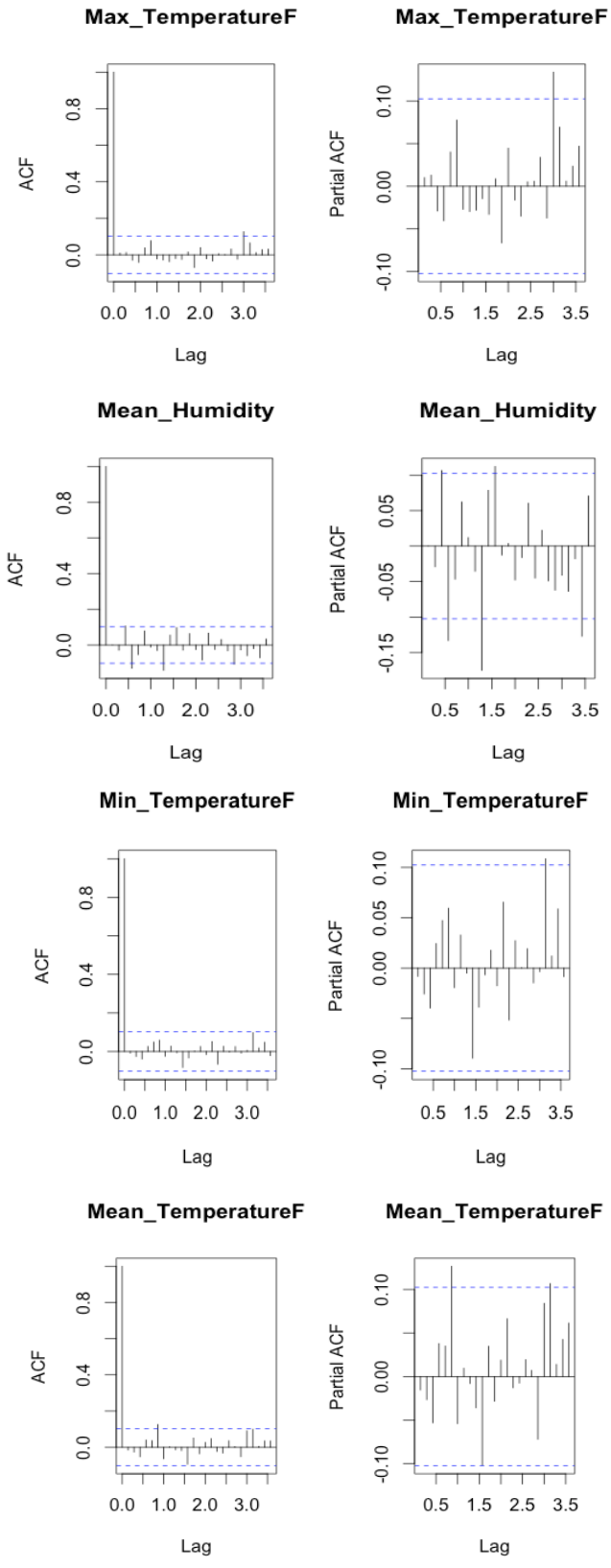

Figure 5 ACF and PACF

We found out that different lags in the ACF and PACF plots in the above figure are within the statistical confidence band, so the selected ARIMA model is adequate.

The final step is then the forecasting of the time series for one or more future time steps ahead. The below plots show that the predicted mean temperature time series is very close to the true value, which indicates again that the ARIMA –model is a valuable tool for short-term forecasting of the variable like the mean temperature, along with prediction of mean temperature at 95% level for future days.

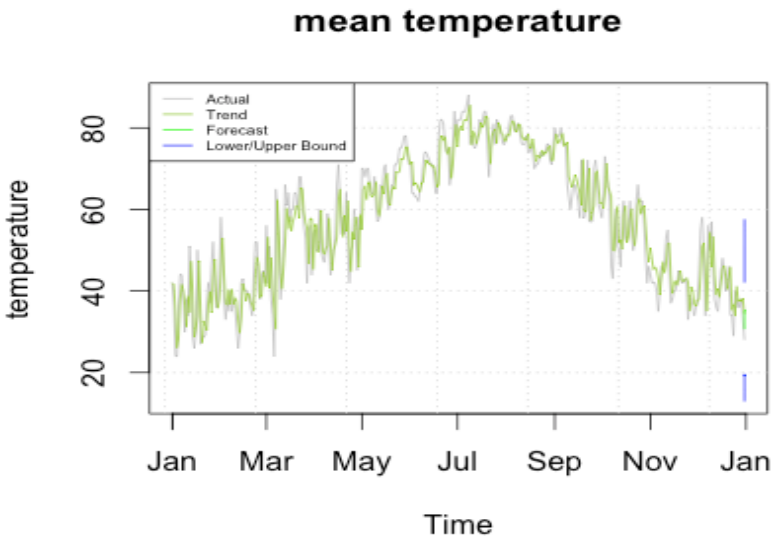

Figure 6 Predicted and Observed Temperature

Conclusion

- Personal water usage is different from west to east America. Consider the shortage of west coast, it is expected to see water usage is under control, while the advanced agriculture and medium population in the middle part lead to high water usage per person.

- To demonstrate current weather data, the data generation method is important. However, the weatherData package tries it best and make it available in R. One problem is that it needs to pre-check data availability, which makes the whole progress quite slow. For instant play, a better data generation tool is needed for improvement.

- Shiny is a great tool to build an interactive plot for those who do not know Java. And the combination power with R makes it great to visualize data. In our case, it is very useful tool to show the temperature information over all countries, and the historical data trend of weather data together with the current local weather. Each mouse action will lead to different data display.

However, it is not as flexible as Java interactive plot. Knowledge of Java and HTML will improve our interactive map a lot.

- The weather data prediction for future days with the ARIMA model suggests that the ARIMA model is a valuable tool for forecasting time series data. Additionally, future temperature forecasting with the ARIMA model also contains the 95% lower and upper bound of the prediction weather value for future days. But when applied in interactive plot, it is quite slow because we use the built-in function. Given the build-in function may be very general to consider every occasions, it may be better for us to write a specific prediction function.