Pittard, William S
Stats290 Final Project Personal Weather Station:

This document supports the Personal Weather Station proposed in the Projects folder on the STATS290 website. For simplicity you may follow the course of action given below to run and evaluate the project.

Note that the only package I used that wasn't already installed on miller1 is the "svMisc" package, which I use to write out a progress bar. I applied the ".onLoad" mechanism described by John Chambers and as far as I can tell it works. However if it does not then svMisc would need to be installed.

Documentation for the functions, classes, and methods are given later in the document and in the code itself, which is contained in a file called "weather.R". There is a script in the "tests" subdirectory of the PWS package folder called "RunMe.R", which can be run in one go with

```
> R --no-save < RunMe.R
```

I) Taking an Interactive Tour

1) Start R

```
> library(PWS)
```

Now create a weather object containing, in this case, info on all PWS within a 5 mile radius of Palo Alto, CA.

```
> myWobj = getWeatherData("Palo Alto,CA",5)
```

```
[1] "Finding lat/lon info for the specified locale(s): Palo Alto,CA"
[1] "Querying the local Geonames database to see whats nearby"
[1] "Checking for locations within 5 miles (approximate) of 
PaloAlto,CA"
[1] "Now checking Wunderground to find stations"
Progress: 49 on 49  Done!
[1] "Found 29 Personal Weather Stations within 5 miles"
[1] "Now checking Wunderground for the station reports"
Progress: 29 on 29  Done!
```

Now use the summary method associated with the weather object

```
> summary(myWobj)
[1] "There are 29 Personal Weather Stations within 5 miles
of PaloAlto,CA"

 BASIC SUMMARY STATISTICS FOR WEATHER STATION ATTRIBUTES
      DIST             ELEV            TEMPF            DEW       
 Min.   :0.250   Min.   :  0.0   Min.   :59.00   Min.   :28.80  
 1st Qu.:2.200   1st Qu.: 49.0   1st Qu.:60.00   1st Qu.:44.00  
 Median :3.570   Median : 70.0   Median :60.30   Median :45.00  
 Mean   :3.723   Mean   :158.8   Mean   :60.67   Mean   :44.59  
 3rd Qu.:5.300   3rd Qu.:190.0   3rd Qu.:61.00   3rd Qu.:46.25  
 Max.   :7.020   Max.   :690.0   Max.   :66.00   Max.   :50.00  
```

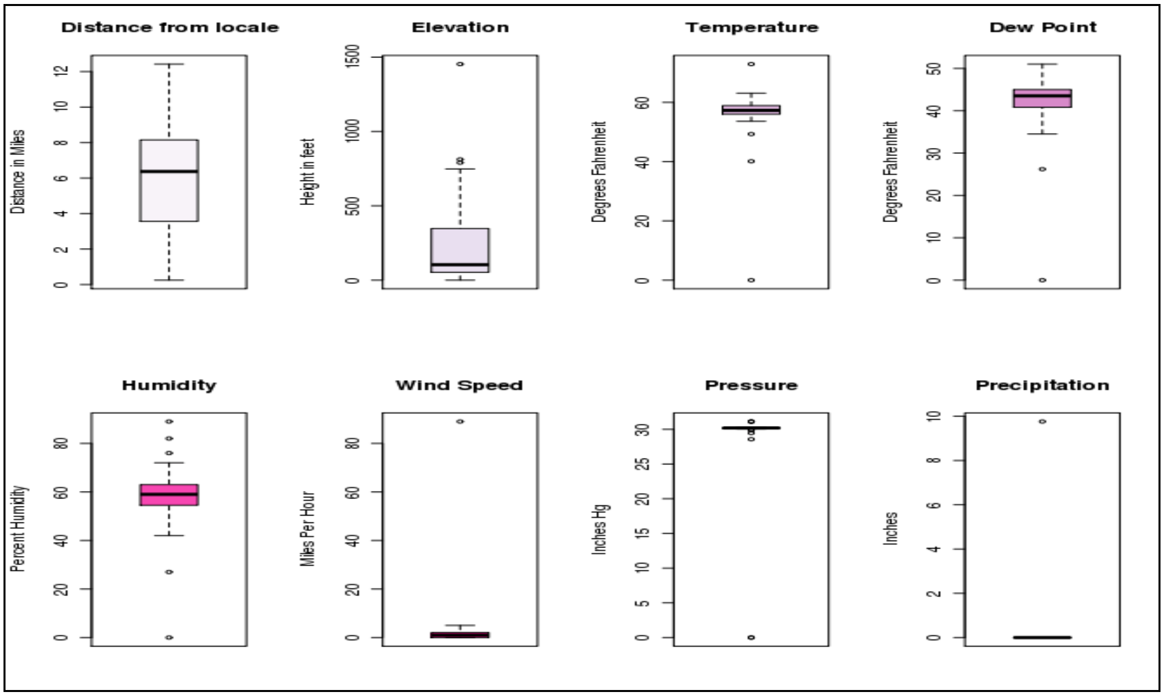
Now use the default “show” method associated with the weather object

```
> show(myWobj)
[1] ">> PWS  report for: PaloAlto,CA Specified radius: 5
miles"

           ID DIST ELEV TEMPF  DEW HUMID W_DIR W_MPH PRESS_IN PRECIP         CITY
1  KCAPALOA11 0.25    7  59.0 46.4    63   ENE   1.0    30.09      0       Palo Alto
2  KCAPALOA18 0.96   36  61.7 46.8    58    SW   0.0    30.11      0       Palo Alto
3  KCAPALOA20 1.38   15  61.8 47.8    60    NE   2.3    30.13      0       Palo Alto
4  KCAPALOA19 1.46   64  60.1 47.5    63 North   3.0    30.13      0       Palo Alto
```

Let’s check out the default plot method

```
>plot(myWobj)
```



The "getLocsByRadius" method will let us query the existing weather object and find all stations within a specified radius of a given station. In this case 1 mile of station KCALOSAL3. Note, the station name must actually exist within the weather object else this won't work.

```
>getLocsByRadius(myWobj,"KCALOSAL3",1)
[1] ">> PWS  report for: PaloAlto,CA Specified radius: 1 miles"

          ID DIST ELEV TEMPF  DEW HUMID W_DIR W_MPH PRESS_IN PRECIP            CITY
12  KCALOSAL1 3.15   96  60.2 50.0    71   NNW   3.3    31.05     NA Los Altos Hills
13 KCALOSAL17 3.48  100  59.5 44.7    58   NNW  89.0    29.99      0       Los Altos
14 KCAMOUNT22 3.56  104  61.3 39.2    51   SSW   7.0    30.16      0   Mountain View
15     MD6539 3.57  347  60.0 45.0    58    NW   2.0    31.04      0       Los Altos
16  KCALOSAL3 3.69  236  61.5 41.1    47   ESE   0.0    29.91      0       Los Altos
17  KCAMENLO7 4.40   70  59.4 44.6    58  West   0.0    30.14      0      Menlo Park
```

Here is an "accessor" method that lets one access data stashed in the weather object. In this case we access column names of the data frame associated with slot "dataDf" of this or any weather object.

```
> getWobjData(myWobj,c("ID","ELEV"))
           ID ELEV
1  KCAPALOA11    7
2  KCAPALOA18   36
3  KCAPALOA20   15
```

```
4  KCAPALOA19   64
5  KCAPALOA13   15
..
..
..
```

Next let’s look at the history methods that allow us to specify a date range and ,for each station in the weather object of interest, obtain historic data. One can create a new object called MyWeatherHist” to accommodate the results. Accordingly there is a default summary method.

```
> dateVec = c(“1/13/11”,”2/21/11”)
> myHist = getHist(myWobj,dateVec)
[1] "Hang on. This could take a while. This isn't part of
the publicized Wunderground API and its a bit slow"

[1] "1/13/11-2/21/11"

> summary(myHist)
[1] "There are 29 stations represented in this history object"

[1] "You might want to use GetHistInfo(WeatherHistObj,\"station\") to
drill down"
       Station            Date          TempHi         TempAvg
 KCAPALOA11: 40   2011-2-16: 29   Min.   :39.0   Min.   :38.00
 KCAPALOA20: 40   2011-2-17: 29   1st Qu.:59.0   1st Qu.:48.00
 KCAPALOA19: 40   2011-2-18: 29   Median :63.0   Median :51.00
 KCAPALOA13: 40   2011-2-19: 29   Mean   :63.1   Mean   :51.39
 KCAMOUNT15: 40   2011-2-20: 29   3rd Qu.:69.0   3rd Qu.:54.00
 KCAPALOA9 : 40   2011-2-21: 29   Max.   :88.0   Max.   :71.00
 (Other)   :838   (Other)  :904
```

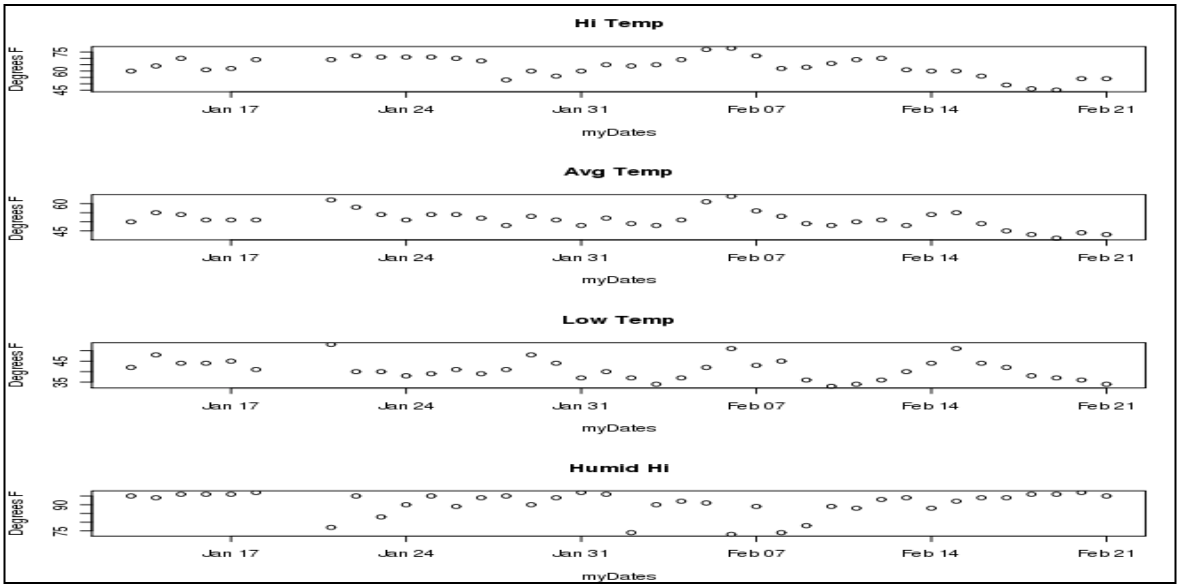
There is also a method called “GetHistData” that will let us get specific information for a particular station out of the History object.

```
> myStationHist = getHistInfo(myHist,"KCAPALOA15")
> myStationHist
[1] ">> PWS  history report for date range 1/13/11-2/21/11"

       Station      Date TempHi TempAvg TempLow DewPtHi DewPtAvg
DewPtLow HumidHi HumidAvg HumidLow PressMax PressMin
278 KCAPALOA15 2011-1-13     60      50      42      54       47
40       95       89       73    30.39    30.33
279 KCAPALOA15 2011-1-14     64      55      48      51       49
46       94       80       58    30.41    30.31
280 KCAPALOA15 2011-1-15     70      54      44      50       47
42       96       80       44    30.33    30.21
```

```
281 KCAPALOA15 2011-1-16      61       51       44       50        47
43       96        88        65    30.28    30.20
..
..
```

```
 >plot(myStationHist)
```



One can also specify multiple cities in the original query. In such cases the first city will be the "reference" city. This is important when computing and reporting distances.

```
> myWobj = getWeatherData(c("Atlanta,GA","Athens,GA"),5)
[1] "Finding lat/lon info for the specified locale(s): Atlanta,GA"
[2] "Finding lat/lon info for the specified locale(s): Athens,GA"
[1] "Querying the local Geonames database to see whats nearby"
[1] "Querying the local Geonames database to see whats nearby"
[1] "Checking for locations within 5 miles (approximate) of Atlanta,GA"
[2] "Checking for locations within 5 miles (approximate) of Athens,GA"
[1] "Now checking Wunderground to find stations"
Progress: 99 on 99  Done!
[1] "Found 16 Personal Weather Stations within 5 miles"
[1] "Now checking Wunderground for the station reports"
Progress: 16 on 16  Done!
>
```

Of course this will work with international locations as well – *Comme suit:*

```
> myWobj = getWeatherData("Paris,FR",20)
[1] "Finding lat/lon info for the specified locale(s): Paris,FR"
```

```
[1] "Querying the local Geonames database to see whats nearby"
[1] "Checking for locations within 20 miles (approximate) of Paris,FR"
[1] "Now checking Wunderground to find stations"
Progress: 49 on 49  Done!
[1] "Found 6 Personal Weather Stations within 20 miles"
[1] "Now checking Wunderground for the station reports"
Progress: 6 on 6  Done!

> myWobj
[1] ">> PWS  report for: Paris,FR Specified radius: 20 miles"

          ID  DIST ELEV TEMPF  DEW HUMID W_DIR W_MPH PRESS_IN PRECIP
CITY
1  IILEDEFR1 10.25  289  48.0 42.1    80   SSE   7.0    29.54   0.01
MASSY
2 IILEDEFR33 14.54  512  48.6 44.6    86   NNE   1.8    29.59   0.00
Fontenay-le-Fleury
3 IILEDEFR20 15.14  541  47.3 43.3    86 North   0.0    29.52   0.00
Voisins-le-Bretonneux
4  IESSONNE6 16.33  312  48.0 43.7    85 North   0.0    29.55   0.00
Longpont-sur-Orge
5     MD1773 18.98  370  49.0 33.0    54    SE   4.0    29.60   0.00
Trappes
6 IILEDEFR53 20.18    0  47.4 43.7    87   SSE   4.0    29.60   0.03 Le
Mesnil-Saint-Denis
```

II) Comments

GeoCoding – I spent most of my time coming up with the best way to GeoCode lats/lons before eventually deciding on implementing a local MySQL copy of the GeoNames database. The free service at GeoNames is pretty bad as it limits requests (unless you pay) and allows only a limited radius. The local version of the database is much more flexible and reliable And returns lots of locations (perhaps too many) so I had to limit the types of locales I got when do an initial search for locations with a given radius. For this reason I excluded restaurants, hotels, shopping malls and stuck with cities, governments, schools, and neighborhoods. Here is the schema I used to accommodate the local database. Of course this is present on miller1.

```
`geo_id`               INT(11) UNSIGNED NOT NULL PRIMARY KEY,
`geo_name`             VARCHAR(200) NOT NULL DEFAULT '',
`geo_ansiname`         VARCHAR(200) NOT NULL DEFAULT '',
`geo_alternate_names`  VARCHAR(2000) NOT NULL DEFAULT '',
`geo_latitude`         DOUBLE PRECISION(11,7) NOT NULL DEFAULT '0',
`geo_longitude`        DOUBLE PRECISION(11,7) NOT NULL DEFAULT '0',
`geo_feature_class`    CHAR(1) ,
`geo_feature_code`     VARCHAR(10) ,
`geo_country_code`     CHAR(2),
`geo_country_code2`    VARCHAR(60),
`geo_admin1_code`      VARCHAR(20) DEFAULT '',
`geo_admin2_code`      VARCHAR(80) DEFAULT '',
`geo_admin3_code`      VARCHAR(20) DEFAULT '',
`geo_admin4_code`      VARCHAR(20) DEFAULT '',
`geo_population`       BIGINT(11) DEFAULT '0',
`geo_elevation`        INT(11) DEFAULT '0',
`geo_gtopo30`          INT(11) DEFAULT '0',
`geo_timezone`         VARCHAR(40),
`geo_mod_date`         DATE DEFAULT '0000-00-00'   ) CHARACTER SET utf8
```

Graphics – Because I spent most of my time on Geocoding and setting up functions and objects the resulting plots look a little drab. This was simply because of time constraints. Obviously some better plots and maps could be generated (e.g. Fusion Tables, ggplot).

II) Functions

1) TITLE: getWeatherData(locale,distance,statLimit)

INPUT: Location in character format e.g. "Atlanta,GA" or a vector with such information such as c("Palo Alto,CA"). statLimit exists to limit the number of possible weather stations that could be returned.

OUTPUT: A S4 weather object containing information on Personal Weather Stations within the specified radius

This is the user interface to get the information into an object that can later be queried.

2) TITLE: deg.dist(long1, lat1, long2, lat2)

INPUT: An originating point and a distal point both specified in lat/lon degrees

OUTPUT: A distance in miles between the two points

3) TITLE: getLatLon(locale, dist, locLimit)

INPUT: Location in character format e.g. "Atlanta,GA" or a vector with such information (e.g. c("Palo Alto,CA")

OUTPUT: A data frame with locations within the specified radius

This function takes a city/state/country string (or vector of such strings) and a distance in miles. It then uses this information to query Google's geocoding service, which is pretty good about returning sane information. **NOTE:** This function is not intended to be called directly by a user as it supports the getWeatherData function, which is the preferred user interface.

4) TITLE: getPlaces(lat, lon, distance, locLimit)

INPUT: A latitude and longitude, a radius/distance, and a limit on how many locations to return

OUTPUT: A dataframe of locations falling with the given radius and the associated lat/lon info

NOTE: In the typical use case this function is not meant to be called directly by the user. It is usually called only by getLatLon(). This function accepts a latitude, longitude, distance (in miles), and a limit on how many stations to return. It returns a data frame containing locations of significance as returned by a local MySQL copy of the main Geonames service.

5) TITLE: getStations(lldf, radius, numofStatLimit)

INPUT: A dataframe ,as returned by getLatLon, which contains lat/lon locations within the given radius. The number of stations returned can be limited to "numofStatLimit". This exists since for a given radius some areas are far more dense with PWS than others - "PaloAlto,CA" vs. "Billings,MT"

OUTPUT: A list of Personal Weather Station ids for which we wish to retrieve weather information

This function accepts a data frame, extracts the lat/lon information and builds a query string suitable for use with the Wunderground weather service to return a list of Personal Weather Station Identifiers.

WARNING: This function could take a while to run since it is dependent on how many queries are given to Wunderground as well as the response time of that site - They did not publish access limits on their API Wiki but I'm sure they have them.

6) TITLE: getWeather(slist,lat,lon,origRadius,baseStation)

INPUT: A list of personal weather station identifiers and the lat/lon of the reference city ,(the first city name specified to getLatLon). Also takes the radius specified in the original called to GetLatLon() so we can trim out any results for stations that might be over the specified radius. Note this won't be many - just a few outliers.

OUTPUT: A S4 oject that has weather information for all the PWS identifiers. It also includes the distance that each station is from the original/reference city
This function's job is to go to the Wunderground site with the station list and obtain weather statistics for each. We'll load this info into a dataframe, do some data checks, and then load up the S4 weather object to hold this data for subsequent interrogation.

II) Classes

1) TITLE: myWeatherObj

Purpose: S4 class to hold information on personal weather stations found within a specified radius:

```
setClass("myWeatherObj",
          representation(origin = "vector",
                         radius = "numeric",
                         baseStat = "vector",
                         dataDf = "data.frame"),
                         contains=c("list","data.frame")
                         )
```

Supporting methods include:

Method	**Purpose**
SetValidity	Validate new myWeather objects
show	Default display method for objects of this class
summary	A default summary method for this class
Plot	A default plot method for this class
getWobjData	An accessor Method for this object
getLocsByRadius	Given a myWeather object – this is a method to locate personal weather stations within a specified radius
getHist	A method to obtain historic data for stations in a weather object for a given date range

1) TITLE: myWeatherHist

Purpose: S4 class to hold history information on personal weather stations contained in a “myWeatherObj” object

```
setClass("myWeatherHist",
                         representation(timeframe = "vector",
                         baseStat = "vector",
                         dataDf = "data.frame"),
                         contains=c("list","data.frame")
                         )
```

Supporting methods include:

Method	**Purpose**
show	Default display method for objects of this class
summary	A default summary method for this class
plot	A default plot method for this class
getHistInfo	An accessor Method for this object