**Running the web service bottlemet.py:**

**Server:**

1. Install bottle *(sudo easy\_install bottle)* and numpy *(sudo easy\_install numpy)*
2. Import rcmes files *(svn export http://oodt/repo/projects/rcmes)*
3. Run web service *(python bottlemet.py)*

**User:**

There are two ways for a user to go through this project, one enabling interactive online processes, and the other directed towards command line users.

Instructions specific to online users are entered in green.

Instructions specific to command line users are entered in orange.

Black instructions apply to all users.

1. Create an instance of class MetricWebService() with the added variable of the desired metric
   1. Online Users:
      1. Users must visit http://localhost:8080/rcmet/metrics/online
      2. They then select the link that corresponds to the desired metric
      3. They will be redirected to a new page that creates an instance of the class and asks the user to choose how to submit variables. A user may choose between posting command line forms and querying the RCMED database. The user selects the link of the method they wish to use. If they have multiple variables, they can submit them in different ways.
   2. Command Line Users:
      1. Users send a form to the URL *http://localhost:8080/rcmet/metrics*
      2. In this form, they post a variable entitled “NameOfMetric”, and set the value as one of the metric names, in string format
      3. Example with curl:
      4. *curl –F NameOfMetric= “calc\_bias”* [*localhost:8080/rcmet/metrics*](http://localhost:8080/rcmet/metrics)*/commandline*
      5. After creating an instance, Bottle automatically proceeds to run a function that allows the user to post variables.
   3. **Bottle creates an instance of the class *MetricWebService().* This instance is always named *BottleMetrics*.**
   4. Available metrics:
      1. metrics with one variable:
         1. calc\_stdev
      2. metrics with two variables:
         1. calc\_annual\_cycle\_means
         2. calc\_annual\_cycle\_std
         3. calc\_annual\_cycle\_domain\_means
         4. calc\_annual\_cycle\_domain\_std
         5. calc\_bias
         6. calc\_bias\_dom
         7. calc\_difference
         8. calc\_mae
         9. calc\_mae\_dom
         10. calc\_rms
         11. calc\_rms\_dom
         12. calc\_temporal\_pat\_cor
         13. calc\_pat\_cor
         14. calc\_anom\_cor
         15. calc\_nash\_sutcliff
         16. calc\_pdf
      3. metrics with three variables
         1. calc\_anom\_cor
2. The user has a selection of methods to choose from, having created an instance of class *MetricWebService()*, and may use some such methods at this time.
   1. To access any of the methods, the user from the command line needs to post the name of the method (without *MetricWebService* in it, of course) in a form entitled MethodName. The value of this form must be a string. When a user makes this post, they must post to http://localhost:8080/rcmet/metrics/commandline/methods. Doing so will run the method, which means it will either return for the user some information on the metric (it also prints this information) or else perform the processes listed under the metric, as in for CleanUp().
   2. An online user will have the option to Run Methods constantly returned as a link on all web pages. Clicking this link will take them to another page that displays the results of running that method. The results will also be printed. Only certain methods are available to the user in this way.
   3. Methods
      1. MetricWebService.Status()
         1. returns a description of the metric and informs the user how many variables are posted, how many are needed, which variables will be represented by which posted values, etc.
      2. MetricWebService.ExplainMetric()
         1. returns a basic description of the metric, obtained from its docstring.
      3. MetricWebService.VariableCount()
         1. this method is called automatically by bottle, and counts the variables submitted. If all variables have been submitted, the function automatically runs MetricWebService.RunMetrics()
      4. MetricWebService.ProcessVariables(array, ArrayName)
         1. this method calls to MetricWebService.VariableCount() after having appended new arrays and their names to the appropriate lists. This method also increases the count to indicate added values.
      5. MetricWebService.GetVariablesFromCommandLine()
         1. A method used whether a user submits variables working from terminal or online. This method requests values from forms, and unpickles them. It then runs MetricWebService.ProcessVariables.
      6. MetricWebService.GetVariables()
         1. this method is run by Bottle *only when a user is working from the command line*, and merely runs two separate functions that set up POST requests at two different URIs depending on whether the user submits data directly or queries RCMED. This method runs either MetricWebService.GetVariablesFromCommandLine() or MetricWebService.ProcessVariables depending on whether variables have been submitted from the command line or through RCMED
      7. MetricWebService.RunMetrics()
         1. the program will run this method automatically once all variables have been submitted. This method returns a string of the result from the metrics. To make this string into something useable in, say, a function, one would have to read() it. See notes below, number 7.
      8. MetricWebService.ReturnResult():
         1. Links the result to a URL, where a command line user can easily fetch it with, say, python module urllib. The URL from which the user would get the result is http://localhost:8080/rcmet/metrics/commandline/return/result
      9. MetricWebService.CleanUp(*name*)
         1. empties the lists ArrayNames and ListOfArrays
         2. returns count to zero
         3. re-assigns the MetricName variable as *name*
         4. Basically, after running MetricWebService.CleanUp(NewMetricName), a user could proceed to run another metric with the same class instance. This function is directed primarily at command line users, as online users may simply return to the first page and start over. However, online users may run this function on the command line and then proceed to other webpages.
3. The user submits variables, as many as a metric requires. This can be done either by posting the variables themselves as arrays, directly from the command line, or by posting a series of parameters that will them be used to search the RCMED database
   1. For a command line user:
      1. To post arrays directly, submit two forms, one with the name *array* and the other with the name *name*, to http://localhost:8080/rcmet/metrics/get/variables/commandline
      2. Assign *array* the value of the variable you wish to submit for the metric
         1. The value assigned to *array* should be in the form of a serialized file or serialized string. It will be unpickled
      3. Assign *name* a value that briefly names or describes your array.
         1. The value assigned to *name* should be a short string
      4. To query the RCMED database, users post 10 forms to http://localhost:8080/rcmet/metrics/get/variables/rcmed, in order to start a query.
         1. *datasetID*
         2. *paramID*
         3. *latMin* attached to a float value
         4. *latMax* attached to a float value
         5. *lonMin* attached to a float value
         6. *lonMax* attached to a float value
         7. *startTime* attached to a datetime object value
         8. *endTime* attached to a datetime object value
         9. *cachedir* attached to a string value
         10. *ArrayName* attached to a string value
      5. Querying RCMED will return multiple values, but this program uses only mdata
   2. For an online user
      1. Depending on whether the user selected the link for directly posting arrays or the link for querying RCMED, they will be redirected to a page with instructions for them.
      2. Using curl, the user may post two forms, *array* and *name,* to /rcmet/metrics/online/commandline
      3. *array* must be a serialized string and *name* must be a string
      4. To query RCMED, the user fills out the appropriate form boxes on the webpage and click submit. Querying RCMED will return multiple values, but this program uses only mdata.
   3. The user should receive a short print message saying a variable was received or found in RCMED, and they will then either be prompted to submit another variable, or else the program should start to run the metric, in which case “Running metric” will be printed, and the final value will be returned
      1. Command line users: to submit another variable, a user may simply post again. They will be notified if this is necessary through a printed message
      2. Online users: to submit another variable, they must on the link they want
         1. *Online*, *Command Line*, or *No More Variables*
   4. Use any of the class methods to manipulate the result.
4. Running the Metrics:
   1. For both users, the metrics will run on their own once all variables have been submitted. Command line users may use the appropriate methods to print or return their result. Online users may use the same methods, but they will also see the result at http://localhost:8080/rcmet/metrics/online/calculate. This page will also contain the option of returning to the first page
5. Potential Problems that could arise.
   1. The types of the datasetID and paramID in the query for RCMED were unclear. They may need to be filled in.
   2. All the documentation instructs the user to go to URLs with localhost:8080 in them. This may need to be changed.
6. Other information
   1. One way to turn a string from a URL into a useable variable:
      1. import urllib
      2. getting\_string=urllib.urlopen(‘http://localhost:8080/…’)
      3. variable=getting\_string.read()
   2. This will change the string of the result in ReturnResult() into a **string** that a program can interact with.
   3. If the result of the metrics is an array, I would recommend including a provision in ReturnResult() that pickles or somehow serializes the result, and then, upon getting the pickled string from the URL, the user could unpickle it and use it as an array.

-------

**Installing cURL:**

1. *sudo easy\_install curl*

**Notes on cURL:**

1. cURL is a framework that allows the user to interact with a server simply by using the command line.
2. HTTP 1.1 protocol is assumed unless otherwise specified
3. For this web service, the most important command is:
   1. *curl -F name=value http://web.address/url…*
   2. This sends a POST request entitled 'name' and containing 'value' to a form at the specified URL
   3. -F can also be written as –form
   4. The http:// part of the web address is optional, as curl assumes HTTP 1.1.
4. The web service will ask for multiple forms, i.e. 'array' and 'name'. These would be submitted:
   1. *curl -F array=variable\_for\_array -F name=name\_of\_array http://web.address/url*
5. For help:
   1. *curl -h*
6. To download a file from a page (or to see html script for a page):
   1. *curl http://web.address/url…*
7. For any other commands, though none should be necessary for this Web Service:
   1. <http://curl.haxx.se/docs/httpscripting.html>