Weather modeling

Swarna Pulavarty

*The University of Montana*

Follow this and additional works at: [https://scholarworks.umt.edu/etd](https://scholarworks.umt.edu/etd)

Let us know how access to this document benefits you.

**Recommended Citation**


[https://scholarworks.umt.edu/etd/7360](https://scholarworks.umt.edu/etd/7360)
Permission is granted by the author to reproduce this material in its entirety, provided that this material is used for scholarly purposes and is properly cited in published works and reports.

**Please check "Yes" or "No" and provide signature**

Yes, I grant permission  

No, I do not grant permission

Author's Signature:  

Date: 8/26/05

Any copying for commercial purposes or financial gain may be undertaken only with the author's explicit consent.
Weather Modeling

by

Swarna Pulavarty

B.Tech, Jawaharlal Nehru Technological University, 2004

presented in partial fulfillment of the requirements

for the degree of

Master Of Science

The University of Montana

Missoula, Montana

December, 2005

Approved by:

Chairperson

Dean, Graduate School

12-20-05

Date

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
Abstract

A program was developed to calculate the difference in weather parameters for observational and forecast data for different locations for different periods. The program gets forecast data from a text file provided by the user and the corresponding observational data from National Weather Service website. The program handles any exceptions raised during retrieval of observational data. Forecast file contains one or more forecast periods.

Observations may be produced for every hour or less than an hour gap, or more than an hour gap. Successive forecast data within a period may differ by 1-hour intervals, up to 24-hour intervals (an extreme case). The two data streams are synchronized for comparison of data. The Observation text file is modified, to produce records consistent with the forecast data interval time gap. The observational data are weighted based on how close they are to the rounding hour. If the observation is near to the rounding hour, it is weighted more than the other. Observations within + / - 60 minutes to the rounding hour are considered.

The software module handles the difference in units for the specified parameters. The forecast data uses Fahrenheit for temperature, Inches for pressure and, Knots for wind speed whereas the observed data uses Celsius for temperature, mbar for pressure and mph for wind speed. It converts the observation data into forecast units and, calculates the difference in corresponding values. It provides the user the flexibility to change from one unit to another in case needed.

Based on user’s input it either plots graphs or creates a summary file with text data. If graphical output is selected by the user, it plots graphs for observational and forecast data for temperature, pressure, wind speed and wind direction. Saves the files as .png, creates a html file with a table of data, and graphs. It gets a copy of the existing index.html file from the remote server to local machine and updates it to reflect changes. The index.html file contains a table of links to the previously run forecast files. These files are present on the server in WebOutput_Files directory. If text output is selected by the user, it creates an output summary file for each forecast file and uploads it onto the server in SummaryOutput_Files directory.
Acknowledgements

I would like to thank my project advisor, Dr. Don Morton, for generating the idea for this project and for his help, advice and support. I feel extremely grateful for the opportunity to spend a semester working on a project in a field of great personal interest under his direction. My project work is part of his Research work being done in Weather Modeling. Dr. Morton is a true expert in the field of Networking, Parallel Processing as well as being a devoted teacher to his students.

I would also like to thank Dr. Joel Henry for his key advice concerning the design aspect of the project. His advices made me do the work in an organized way which made my implementation task easy. His classes on software engineering were very useful in getting me up to speed with the design required for this project. He took time from his busy academic schedule to respond to my enquiry in a very helpful and detailed fashion. This advice proved essential to whatever small success this project may have achieved.

I would also like to thank Dr. George McRae for his valuable advice concerning different aspects of the project. His initial guidance regarding the project has helped me visualize the different aspects and requirements of a project.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABSTRACT</td>
<td>ii</td>
</tr>
<tr>
<td>ACKNOWLEDGEMENTS</td>
<td>iii</td>
</tr>
<tr>
<td><strong>CHAPTER 1</strong></td>
<td></td>
</tr>
<tr>
<td>Introduction (Background work and Purpose)</td>
<td>1</td>
</tr>
<tr>
<td>Project Organization</td>
<td>2</td>
</tr>
<tr>
<td><strong>CHAPTER 2</strong></td>
<td></td>
</tr>
<tr>
<td>Development Environments Used</td>
<td>3</td>
</tr>
<tr>
<td>Prerequisites</td>
<td>3</td>
</tr>
<tr>
<td>Initial Assumptions</td>
<td>4</td>
</tr>
<tr>
<td><strong>CHAPTER 3</strong></td>
<td></td>
</tr>
<tr>
<td>Retrieving Forecast and Observational data</td>
<td>6</td>
</tr>
<tr>
<td>Synchronizing Forecast and Observational data</td>
<td>8</td>
</tr>
<tr>
<td>Naming files and Uploading onto the Server</td>
<td>9</td>
</tr>
<tr>
<td><strong>CHAPTER 4</strong></td>
<td></td>
</tr>
<tr>
<td>Test Cases</td>
<td>11</td>
</tr>
<tr>
<td>Man Pages</td>
<td>14</td>
</tr>
<tr>
<td>User Comments on Prototype Model</td>
<td>19</td>
</tr>
<tr>
<td><strong>CHAPTER 5</strong></td>
<td></td>
</tr>
<tr>
<td>How is the Program presented to User</td>
<td>21</td>
</tr>
<tr>
<td>Conclusions</td>
<td>21</td>
</tr>
<tr>
<td>How to Make Future Changes ?</td>
<td>21</td>
</tr>
<tr>
<td>Future Work</td>
<td>23</td>
</tr>
<tr>
<td><strong>REFERENCES</strong></td>
<td>24</td>
</tr>
</tbody>
</table>
CHAPTER 1

Introduction

Background

The National Weather Service is a component of the Operating Unit of National Oceanic and Atmospheric Administration (NOAA) of the U.S. Department of Commerce. The National Weather Service (NWS) provides weather, hydrologic, and climate forecasts and warnings for the United States, its territories, adjacent waters and ocean areas, for the protection of life and property and the enhancement of the national economy. NWS data and products form a national information database and infrastructure which can be used by other governmental agencies, the private sector, the public, and the global community. My project gets the Observational data from this site.

The Weather Research and Forecasting (WRF) Model is a next-generation mesoscale numerical weather prediction system designed to serve both operational forecast and atmospheric research needs. Research work of my professor Don Morton, is to capture the effects of rugged topography on local weather in daily numerical weather forecasts relating to Aviation, Firefighting and Research and to test the ability of WRF (and improve upon) in capturing the real-world weather in such regions.

Currently, WRF is setup to run on the domains of interest on different parallel architectures. The process is automated to produce a daily forecast, from fetching the initialization data, to pre-processing, to job queuing, to post-processing and web posting.
Forecast data for my project is obtained from the weather model, WRF which is archived onto the data server, Snowy. This system provides open access to data useful to Alaskans and others interested in high latitudes. It is intended for providing local access to global resources, by mirroring useful software and content from around the world.

**Purpose**

The primary purpose of my project is to implement robust software that gets Forecast data from the user (user gets it from the WRF Model) in the form of a text file and gets the observed meteorological data from the National Weather Service website that matches the Forecast data for the specified location, for the specified number of hours programmatically. It then, synchronizes both of them and compares the data in meaningful ways, and presents it in a logical, user-friendly format.

**Project Organization**

The rest of the project is organized as follows.

Chapter 2 mentions the development environments used, prerequisites and the initial assumptions taken to run the program.

Chapter 3 describes the method used to accomplish the goal of the project.

Chapter 4 specifies the test cases run to test the robustness of the program, man pages to guide the user run the program and user comments on prototype model.

Chapter 5 specifies the conclusions, future work, and how the user can make some of the future changes.
CHAPTER 2

Development Environments used

This project is developed on a Red Hat Linux machine. The following software’s are used. Python-2.4.2, Gnuplot-4.0.0, gnuplot-1.7.1, Numeric Python, HTMLgen and PIL (Python Imaging Library). The program is written in python using necessary modules.

Prerequisites

The system, on which the program is to be run, should have Red Hat Linux installed on it, with 2.4.21-37.ELsmp kernel release and SMP kernel version that uses i686 processor.
Initial Assumptions

- The program assumes that the observational data is obtained from the URL provided by the National Weather Service website and can be changed by the user at any later time.
- The program assumes that the user gives the path of the forecast file in the local system.
- The program assumes that the forecast file has one or more forecast Periods.
- The program assumes that successive records have the same interval gap in a forecast period and are multiples of hours.
- The program assumes that interval gaps between successive records may vary across forecast periods.
- The program assumes that the forecast periods are separated by some sort of common delimiters.
- The program assumes that all forecast periods in a forecast file belong to a same location.
- The program assumes that observational data exists for any given forecast data in the forecast file.
- The program assumes that a remote directory with the Name "WebOutput_Files" exists on the webserver for graphical output.
- The program assumes that a remote directory with the Name "SummaryOutput_Files" exists on the webserver for text output.
• The program assumes that an index file initially exists with an empty table to hold graphical data. It is inside the WebOutput_Files directory on the remote webserver.

• The program assumes that the browser supports .png files.

• The program assumes that the user runs this program on red hat linux machine.
CHAPTER 3

Retrieving Forecast data and Observational data

The forecast data is retrieved from a file using the location provided by the user on the local machine. The Forecast data from the model may be for 12 hours or 24 hours, or 48 hours or 72 hours or 96 hours or for that matter, for any number of hours with arbitrary time interval gap between forecast periods (in multiples of hours). Each forecast file has one or more forecast periods. Successive records in a forecast period have the same interval gap and are in multiples of hours. Forecast periods are separated by some sort of common delimiter inside a forecast file. All the forecast periods in a forecast file belong to a same location. For each forecast period in a forecast file, the program gets the corresponding observation data, synchronizes both of them, performs the required operation, and uploads them onto a web server.

The program calculates the number of hours required, to retrieve observational data for the forecast period using python's time module. It calculates the number of seconds lapsed from the epoch to the forecast timestamp, and the number of seconds lapsed from the epoch to the current system time. It then calculates the difference of the two and converts it into hours. For each forecast period in a forecast file the corresponding observational data is retrieved and saved to a file.

The observational data is retrieved from the National Weather Service, Western Region Headquarters website http://www.wrh.noaa.gov/mesowest/getcsv.php for the calculated number of hours. It provides the user, the flexibility to change the URL, if needed. Any errors encountered during the retrieval of data, are handled, rather than abnormally terminating the program. Possible errors handled are Page not found, Internal Server
Error, No Data Available, and Page Not Modified [1]. If data could not be retrieved, then the software handles it, such that, it makes attempts to retrieve it again after some time. It also handles page redirects (Temporary and Permanent redirects). In case a permanent redirect occurs, it makes sure that it uses the new URL, from the next time to retrieve data. If it is unable to recover from the errors, it gives some descriptive output to help the user determine what might have gone wrong.

Observations may be produced for every hour or less than an hour gap, or more than an hour gap. Successive forecast data within a period may differ by 1-hour intervals, up to 24-hour intervals (an extreme case). The two data streams should be synchronized for comparison of data. The Observation text file should be modified, to produce records consistent with the forecast data interval time gap. The observational data are weighted based on how close they are to the rounding hour. If the observation is near to the rounding hour, it is weighted more than the other. Observations within +/− 60 minutes to the rounding hour are considered.

The software module handles the difference in units for the specified parameters. The forecast data uses Fahrenheit for temperature, Inches for pressure and, Knots for wind speed whereas the observed data uses Celsius for temperature, mbar for pressure and mph for wind speed. It converts the observation data into forecast units and, calculates the difference in corresponding values. It provides the user the flexibility to change from one unit to another in case needed.

Sample format of the Forecast file is at Appendix A.
Synchronizing Forecast and Observational data

As seen in Appendix B, the observation data may be produced for every hour or for less than an hour if sudden changes occur or for more than an hour if the weather is static. These observations are modified to represent data consistent with the forecast data interval time gap. Normalization of observational data is done using the formula

\[
\frac{\Delta T_{\text{extreme}} - \Delta T_{\text{up}}}{\Delta T_{\text{extreme}}} \times \text{Param}_{\text{up}} + \frac{\Delta T_{\text{extreme}} - \Delta T_{\text{up}}}{\Delta T_{\text{extreme}}} \times \text{Param}_{\text{down}}
\]

where

\[
\Delta T_{\text{extreme}} = T_{\text{down}} - T_{\text{up}} = \text{time difference between the two observation records}
\]

\[
\Delta T_{\text{up}} = T_{\text{round}} - T_{\text{up}} = \text{time difference between the rounding hour and the upper observation record.}
\]

\[
\Delta T_{\text{down}} = T_{\text{down}} - T_{\text{round}} = \text{time difference between the rounding hour and the lower observation record.}
\]

Param = parameter under consideration (one of temperature, pressure, wind speed, wind direction)

Records nearest to the rounding hour are considered. Rounding hour is the timestamp in the forecast period. Records with less than +/- 60 minutes difference between rounding hour and the observation hour are considered for estimation. If the difference is greater than +/- 60 minutes then such records are neglected, and a no estimate value is assigned to the observational parameter at that rounding hour.

On normalizing the observation records at Appendix B, they look as
<table>
<thead>
<tr>
<th>Timestamp</th>
<th>ObsTemp</th>
</tr>
</thead>
<tbody>
<tr>
<td>2005120500</td>
<td>-18.89</td>
</tr>
<tr>
<td>2005120421</td>
<td>-20.0</td>
</tr>
<tr>
<td>2005120418</td>
<td>-20.0</td>
</tr>
<tr>
<td>2005120415</td>
<td>-21.11</td>
</tr>
<tr>
<td>2005120412</td>
<td>-24.44</td>
</tr>
<tr>
<td>2005120409</td>
<td>-23.33</td>
</tr>
<tr>
<td>2005120406</td>
<td>-23.33</td>
</tr>
<tr>
<td>2005120403</td>
<td>-22.22</td>
</tr>
<tr>
<td>2005120400</td>
<td>-18.89</td>
</tr>
<tr>
<td>2005120321</td>
<td>-16.11</td>
</tr>
<tr>
<td>2005120318</td>
<td>-17.78</td>
</tr>
<tr>
<td>2005120315</td>
<td>-17.78</td>
</tr>
<tr>
<td>2005120312</td>
<td>-19.44</td>
</tr>
<tr>
<td>2005120309</td>
<td>-18.89</td>
</tr>
<tr>
<td>2005120306</td>
<td>-17.22</td>
</tr>
<tr>
<td>2005120303</td>
<td>-17.78</td>
</tr>
</tbody>
</table>

Once the two files are synchronized, for all the parameters under consideration, it calculates the difference in temperature, pressure, wind speed and wind direction and does either one of the following depending on user’s choice.

- Produces a table and graphs for each forecast period that shows the forecast and observational data
- Produces a summary file that shows a table of all the forecasts, observations and the differences.

**Naming files and Uploading onto the Server**

If the user selects graphical output, graphs are plotted for forecast and observational data for each of the parameters using Gnuplot and Python Imaging Library (PIL), and are saved as .png files. A HTML file is generated using HTMLgen module with graphs and a table containing observational data for temperature, pressure, wind speed, wind direction and forecast data for temperature, pressure, wind speed, wind direction and the corresponding differences. The .png files and the html files are named with the starting value of the forecast period, location and parameter under consideration.
The output graphs, the observational and forecast data for that period should be uploaded onto the web server in user’s account. For this, a connection is made to the remote web server using public encryption method. This public encryption method, allows the user to connect to the remote host, without entering a password each time a connection is made to the remote host. The index.html file (which holds all the previously run links) is copied from the web server to the local machine. A new link is added to the existing index.html at the end of the table to provide access to the newly created html file and the output graphs. Then, all the files, i.e the png files, html file, output summary file and the updated index.html are uploaded onto the webserver in the WebOutput_Files directory which is assumed to be present in user’s home directory.

When the user visits the index.html file on the webserver, he sees a list of records, with each record specifying the details like the date for which the graph has been plotted, the number of hours, the location, and the starting time for which the graph has been plotted. The date field is a hyperlink that allows the user to click, to view the graphs and data produced for that date.

If the user selects text format for output then a summary file is created to hold all the data in form of a table. The summary text file contains tables with observational, forecast and difference data for temperature, pressure, wind speed and wind direction for each forecast period in the forecast file. Each period is separated by some sort of delimiter. This summary file is uploaded onto the web server in SummaryOutput_Files directory which is assumed to be present in user’s home directory.
CHAPTER 4

Test Cases

Testing for the robustness of the program under url errors or incorrect input or missing forecast data.

Case 1

Prototype testing

The prototype model of this project has been tested with a user unrelated to the project. A requirements document and a design document have been given to the user to go through it, and a couple of questions have been asked to get his opinion. His comments about the prototype were specified under the heading “User Comments on Prototype Model” later in the document.

Case 2

Test the program for cases where status code is 200 but no data is available.

Status code 200 is returned when everything is normal. The program has been run with the url http://www.wrh.noaa.gov/mesowest/getcsv.php?sid=KMSO&num=-1 that gives observational data for the specified location, for the specified number of hours. The number of hours has been changed to -1 and the program is run to see the output. It gave the error that No data is available and retries again after sometime. It made a couple number of tries for an hour and then exited out of the program.

Case 3

Test the program for Page not Found error (status code 404)
The program has been run with a url that always displays the error 'page not found '. It gave the error that page is not found and retries again after sometime. It made a couple number of tries for an hour and then exited out of the program.

Case 4

Test the program for Page Gone error (status code 410)

The program has been run with a url that always displays the error 'page gone '. It gave the error that the page is gone and retried again after sometime. It made a couple number of tries for an hour and then exited out of the program.

Case 5

Test the program for Permanent Redirect (status code 301)

The program has been run with a url that always redirects to another url during retrieval of data. For the first time when the program is run, it redirected the url and retrieved the data correctly and did all the operations. The program is run for the second time to see if its retrieving from the old url or from the new url. It has been observed that it retrieved data from the new url, since all the header and url information have been updated in the first run itself when the status code is 301.

Case 6

Test the program for No permission error (status code 403)

The program has been run with a url that requires username and password for retrieval of data. While running, no username and password were given to the url. It displayed the error message that the page is forbidden and retried again after sometime. It made a couple number of tries for an hour and then exited out of the program.
Case 7

Test the program for Internal Server error (status code 500)

The program has been run with a url that always gives url open error. It displayed the error message that the an error has occurred while opening the url and it retried again after sometime. It made a couple number of tries for an hour and then exited out of the program.

Case 8

Test the program for missing forecast data in the forecast file

A record has been removed from the forecast file and the program is run against this file. It skipped the forecast period that contains missing forecast data, and continued to produce for other forecast periods. It displayed a message that data has been missing at the line no so and so in the forecast file.

Case 9

Test the program for incorrect input from the user

The program has been run by skipping some of the input values. The program exits out of the program, telling the user that some of input values were missing.

Case 10

Test the program for incorrect forecast file path

The program has been run by specifying an incorrect forecast file path. The program checks for the file in the local machine for its existence. If the file isn’t present it reports that the forecast file cannot be found at the specified location and exits.
Man pages

What is the Purpose of this script?

Given a forecast file, with forecast data for different periods, this script gets the corresponding observational data for each forecast period, synchronizes both of them, calculates the difference in temperatures, wind speed, wind direction, station pressure, plots graphs for Forecast and Observational data, or generates a summary file based on user’s input and uploads them onto the web server.

What is the format of the input Forecast file to be provided?

The forecast file should have forecast data for a single period or a couple of periods. If the file has data for a couple of periods, then successive periods should be separated by some sort of delimiter. My script assumes "-----" as delimiter. If the delimiter needs to be changed, Delimiter variable in the Const_Variables.py file should be assigned with the new delimiter.

One limitation with Observational data obtained from the National Weather website, is that, it gives observations only for the past one month. So, it is the user’s responsibility to make sure that observation data exists for the given forecast period.

The script assumes that successive records in the Forecast period differ by the same amount of interval gap. All the periods in the forecast file need not have the same interval gap.

Sample format of the forecast file is at Appendix A

How to run the Python script?

To run the python script on kerlee.cs.umt.edu from a Linux Machine, the command would be similar to
What does runme file contain?

The file runme, is an executable file that contains the executable paths of all the software's used and my MainScript. List of software's used are Python-2.4.4, Gnuplot 4.0.0, gnuplot-1.7, HTMLgen-4.0 and Imaging-1.1.5.

How is the product given to the user?

A tar file is given to the user, which the user needs to untar it, to run the scripts. This tar file contains all the software's used, my scripts, and the runme file. The user should make sure that he is running the runme file from the directory that contains it.

What input arguments need to be given to the Python script at run time?

```
[warnaimcslS5-132 Package]$ ./runme
Bad Location argument

[debug <level>]
-Location (<4 letter code in capitals>)
-ForecastFile (<full path of Forecast file in local machine>)
-OutputFormat (<press 1 for Graphical, press 2 for Summary Output>)
-OutputFile (<type filename with extension if 2 is selected before or else neglect>)
-WebServer (<web server name>)
-WebUser (<user name on the web server mentioned>)
[-help]
```

```
Number of Seconds from epoch to first Forecast timestamp : 1133578800
Number of Seconds from epoch to current system time : 1134414000
Number of Hours : 235
Reading Forecast Data from the File
Retrieving Observational data from the URL
```
Debug level: A number between 1 and 3. 1 implies fewer status messages, 3 implies more number of status messages. 2 is intermediate.

The abbreviation of the Location for which the Observational data is to be retrieved (eg: for Missoula, the abbreviation is KMSO)

The Forecast file Name against which the script has to run.(e.g.Forecast_Data.txt) Make sure that this text file is in the same directory as the python scripts. If not, specify the whole path. Do not forget to mention the file extension. Valid extensions are .txt, .dat, .doc.

Output format which indicates whether a graphical output (press 1) is desired or a summary text file is desired (press 2)

Name for the Output summary text file that contains, the forecast and observation temperatures, corresponding to the forecast input file provided.

The server onto which the output is to be stored.

The username on the server mentioned above.

How to avoid remote host prompting the user for password each time a connection is made?

This can be achieved using public encryption. The user needs to create public and private keys in his local machine and upload the public key onto the remote web server using
using OpenSSH and SSH protocol version 2. For more information click http://www.linuxquestions.org/linux/answers/Jeremys_Magazine_Articles/Using_Keys_with_SSH.

What is the Output produced by running this script?

Depending on the user's choice the output produced will either be a text file with the name provided by the user, or a html file with a table specifying timestamp, observational, forecast and difference data for temperature, pressure, wind speed and wind direction. The html file also contains graphs for forecast and observational data for each of the parameters mentioned above.

Where does the output get stored?

The graphical output gets stored onto the directory "WebOutput_Files" and summary output files gets stored onto the directory SummaryOutput_Files on the server. These directories are assumed to be present in the user's home directory on the web server. These directories can be changed by the user if needed, in the Const_Variables.py file.

How to access the Graphical Output files?

To view the graphical output, the user needs to login into his user account onto the server. The output files are located at /local/home/user-name/WebOutput_files/index.html/

Once the user opens the index.html file, he sees table with a list of records, with details like date link specifying the day for which the graph has been plotted (on clicking takes to the graph), the number of hours, the location, and the starting time for which the graph has been plotted, the interval gap for successive records.

This table also contains links, to the previously run forecast files.
Sample index.html file is at Appendix C.

**How to access the Summary Output files?**

To view the summary output, the user needs to login into his user account onto the server. The summary output file is located at `/local/home/user-name/SummaryOutput_files/`.

Sample Output Summary file is at Appendix D.

**How to interpret the output?**

On clicking one of the date links, in the index.html file, it displays a table with timestamp, forecast and observation data for temperature, station pressure, wind speed and wind direction and, graphs plotted for each of the parameters.

Sample html file output is at Appendix E.
User Comments on Prototype Model

The prototype model of this project has been tested with a user unrelated to the project. A requirements document and a design document has been given to the user to go through it, and a couple of questions have been asked to get his opinion. His comments about the prototype are:

- The user felt that the input arguments to the program were too many. He prefers that the arguments be stored in a text file, and are the default arguments to be supplied to the program at run time. If the user specifies some argument like the forecast file path as a command line argument when running the program, then the program should use the new value given by the user rather than the default value in the text file.

- Since the user doesn't know what arguments to give at the first instance, user prefers that the program should provide some commands similar to Linux commands like -help for help on what arguments need to be given as input to the program.

- User prefers that the constants be placed in a text file rather than a python file, to make it easy for the user to change it at later time.

- The user likes the public encryption method to avoid remote host prompting for password, but has no knowledge of how it can be accomplished. So he prefers to have documentation to accomplish that.

- The user prefers only 2 debug levels, either displaying all the status messages or none. He says that there aren't many status messages that the user needs to know for this project, so he prefers not to have status messages. He also feels its
frustrating to see those status messages for each forecast period especially when the number of forecast periods is more.

- Irrespective of the debug level, the user prefers to see a summary output for operations performed on the forecast periods. If there is some bad forecast data in the forecast file with missing or incorrect format, the user wants the program to keep track of all those and display a summary like

- X of Y forecast periods have succeeded
  line x1 has bad forecast data
  line x2 has bad forecast data
  .
  .
  .

- If the observation data couldn’t be retrieved for some reason, the user prefers that the program exit after a couple number of retries.

- User prefers that the index.html file creates a separate table for each forecast file, with each table containing links to forecast periods in that forecast file. Also prefers to have a link to the Forecast input file one for each table.

- User prefers that on clicking the index.html file (file that maintains the table of records for forecast periods), each html file opened should have a set of links, one for each parameter (like temperature, pressure, wind direction etc.) for which forecast and observation data are compared.

- User felt that the graphical output is clear and simple and doesn’t need any changes.
CHAPTER 5

How is the Program presented to User?

A tar file is created with all the necessary software’s, script files, man pages and a runme executable file. Software’s include Python-2.4.2, Gnuplot-4.0.0, gnuplot-1.7.1, Numeric Python, HTMLgen and PIL (Python Imaging Library). Script files are the files developed by me. Man pages guides the user on how to run the program and how to interpret the output. The runme file contains the executable paths to all the software’s and my scripts. The user needs to untar the package and cd to the untarred package directory from the command line and run the runme executable file.

Conclusions

In this project, we presented an algorithm that retrieves forecast data from the location specified by the user, and observational data from the National Weather Service website, saves to a file, synchronizes both the formats, and calculates the differences in temperature, pressure, wind speed and wind direction. Based on the user’s input it either produces a web output that shows table of data and graphs or a summary text file, which shows only the text data in the form of a table. It then uploads the output onto the user’s home directory on the web server.

How to Make Future Changes?

- To make changes to the URL from where the observational data is retrieved, the user needs to modify the variable “Url” in the Const_Variables.py file. The user can change the header information of the URL like the USER_AGENT.
• To change the delimiter that separates forecast periods, the user needs to change "Seperator" variable in the Const_Variables.py file.

• To change the timeout period for observational data retrieval, the user needs to change the "Timeout" variable in the Const_Variables.py file.

• To include new parameter for comparison, the following files should be changed.
  
  Main_Read.py – An array should be declared to hold the forecast values for that parameter like "ForeParamName". Data value retrieved from the forecast file should be assigned to this array for each record in the forecast period.

  FileEstimate.py – An array should be declared to hold the observation values for that parameter like "ParamName". If the observation value is null then a 0 is assigned to it. Another array is declared to hold the final set of values that are obtained after estimation and conversion of original observation data like "ObsParamName". If there is missing observation data or some hour, a no estimate value is assigned to all the parameters at that hour.

  Compare.py – An array should be declared to hold the difference values of forecast and observational data like with the name "DifferenceParamName". These arrays need to be written to the files, that are used for plotting data and generating summary file. These arrays should be sent as parameters to Generate_HTML.py. .png file name should be created for this new parameter using forecast timestamp, location, and parameter name appended. The graph plotted for forecast and observation data using this parameter is given that name. This filename is sent as a parameter to PlotGraph.py.
Future Work

- To automate the application to run for every 24 hours retrieving the forecast and observational data from arbitrary websites, synchronize them, plot graph for observational and forecast data, and upload them onto the webservice.

- Evaluate the WRF vs. observed comparisons and derive implementation modifications that might improve on forecasts.
### References

1. Dive into Python by Mark Pilgrim.
5. Dr. Don Morton’s sample Python scripts
Appendix A

Forecast File Format

Sample format of the Forecast file would be like:

<table>
<thead>
<tr>
<th>Timestamp</th>
<th>StaPSFC</th>
<th>StaT2</th>
<th>StaWindSpd</th>
<th>StaWindDir</th>
<th>StaRain</th>
</tr>
</thead>
<tbody>
<tr>
<td>2005120300</td>
<td>927.91</td>
<td>13.79</td>
<td>3.35</td>
<td>210.16</td>
<td>0.01</td>
</tr>
<tr>
<td>2005120303</td>
<td>926.54</td>
<td>10.68</td>
<td>4.23</td>
<td>192.95</td>
<td>0.01</td>
</tr>
<tr>
<td>2005120306</td>
<td>927.28</td>
<td>10.74</td>
<td>1.52</td>
<td>163.35</td>
<td>0.01</td>
</tr>
<tr>
<td>2005120309</td>
<td>927.22</td>
<td>9.02</td>
<td>4.67</td>
<td>121.22</td>
<td>0.01</td>
</tr>
<tr>
<td>2005120312</td>
<td>926.54</td>
<td>8.26</td>
<td>9.24</td>
<td>125.45</td>
<td>0.01</td>
</tr>
<tr>
<td>2005120315</td>
<td>925.05</td>
<td>9.85</td>
<td>10.30</td>
<td>134.04</td>
<td>0.01</td>
</tr>
<tr>
<td>2005120318</td>
<td>924.48</td>
<td>11.21</td>
<td>15.36</td>
<td>152.63</td>
<td>0.01</td>
</tr>
<tr>
<td>2005120321</td>
<td>924.20</td>
<td>10.49</td>
<td>10.88</td>
<td>145.11</td>
<td>0.01</td>
</tr>
<tr>
<td>2005120324</td>
<td>923.67</td>
<td>11.11</td>
<td>10.55</td>
<td>136.46</td>
<td>0.19</td>
</tr>
<tr>
<td>2005120327</td>
<td>921.75</td>
<td>8.39</td>
<td>7.69</td>
<td>123.89</td>
<td>0.19</td>
</tr>
<tr>
<td>2005120330</td>
<td>923.31</td>
<td>8.32</td>
<td>3.88</td>
<td>227.38</td>
<td>0.19</td>
</tr>
<tr>
<td>2005120401</td>
<td>925.86</td>
<td>7.91</td>
<td>6.82</td>
<td>217.29</td>
<td>0.19</td>
</tr>
<tr>
<td>2005120404</td>
<td>926.60</td>
<td>6.85</td>
<td>6.53</td>
<td>213.58</td>
<td>0.19</td>
</tr>
<tr>
<td>2005120407</td>
<td>927.14</td>
<td>5.94</td>
<td>7.31</td>
<td>203.70</td>
<td>0.19</td>
</tr>
<tr>
<td>2005120410</td>
<td>929.42</td>
<td>6.16</td>
<td>7.49</td>
<td>216.85</td>
<td>0.19</td>
</tr>
<tr>
<td>2005120413</td>
<td>931.64</td>
<td>6.88</td>
<td>6.27</td>
<td>243.58</td>
<td>0.19</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Timestamp</th>
<th>StaPSFC</th>
<th>StaT2</th>
<th>StaWindSpd</th>
<th>StaWindDir</th>
<th>StaRain</th>
</tr>
</thead>
<tbody>
<tr>
<td>2005090712</td>
<td>963.67</td>
<td>11.11</td>
<td>10.55</td>
<td>136.46</td>
<td>0.19</td>
</tr>
<tr>
<td>2005090713</td>
<td>921.75</td>
<td>8.39</td>
<td>7.69</td>
<td>123.89</td>
<td>0.19</td>
</tr>
<tr>
<td>2005090714</td>
<td>953.31</td>
<td>8.32</td>
<td>3.88</td>
<td>227.38</td>
<td>0.19</td>
</tr>
<tr>
<td>2005090715</td>
<td>965.86</td>
<td>7.91</td>
<td>6.82</td>
<td>217.29</td>
<td>0.19</td>
</tr>
<tr>
<td>2005090716</td>
<td>946.60</td>
<td>6.85</td>
<td>6.53</td>
<td>213.58</td>
<td>0.19</td>
</tr>
<tr>
<td>2005090717</td>
<td>937.14</td>
<td>5.94</td>
<td>7.31</td>
<td>203.70</td>
<td>0.19</td>
</tr>
<tr>
<td>2005090718</td>
<td>979.42</td>
<td>6.16</td>
<td>7.49</td>
<td>216.85</td>
<td>0.19</td>
</tr>
<tr>
<td>2005090719</td>
<td>981.64</td>
<td>6.88</td>
<td>6.27</td>
<td>243.58</td>
<td>0.19</td>
</tr>
<tr>
<td>2005090720</td>
<td>967.64</td>
<td>6.88</td>
<td>6.27</td>
<td>243.58</td>
<td>0.19</td>
</tr>
<tr>
<td>2005090721</td>
<td>938.64</td>
<td>6.88</td>
<td>6.27</td>
<td>243.58</td>
<td>0.19</td>
</tr>
<tr>
<td>2005090722</td>
<td>962.64</td>
<td>6.88</td>
<td>6.27</td>
<td>243.58</td>
<td>0.19</td>
</tr>
<tr>
<td>2005090723</td>
<td>934.64</td>
<td>6.88</td>
<td>6.27</td>
<td>243.58</td>
<td>0.19</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Timestamp</th>
<th>StaPSFC</th>
<th>StaT2</th>
<th>StaWindSpd</th>
<th>StaWindDir</th>
<th>StaRain</th>
</tr>
</thead>
<tbody>
<tr>
<td>2005090800</td>
<td>933.79</td>
<td>8.25</td>
<td>6.48</td>
<td>243.53</td>
<td>0.30</td>
</tr>
<tr>
<td>2005090803</td>
<td>934.74</td>
<td>7.45</td>
<td>5.59</td>
<td>255.11</td>
<td>0.30</td>
</tr>
<tr>
<td>2005090806</td>
<td>938.22</td>
<td>7.21</td>
<td>6.47</td>
<td>268.90</td>
<td>0.30</td>
</tr>
<tr>
<td>2005090809</td>
<td>940.87</td>
<td>7.03</td>
<td>5.60</td>
<td>255.21</td>
<td>0.30</td>
</tr>
<tr>
<td>2005090812</td>
<td>943.24</td>
<td>5.19</td>
<td>12.53</td>
<td>249.75</td>
<td>0.30</td>
</tr>
<tr>
<td>2005090815</td>
<td>944.79</td>
<td>3.79</td>
<td>9.09</td>
<td>253.17</td>
<td>0.30</td>
</tr>
<tr>
<td>2005090818</td>
<td>946.88</td>
<td>6.07</td>
<td>8.18</td>
<td>255.10</td>
<td>0.30</td>
</tr>
<tr>
<td>2005090900</td>
<td>948.55</td>
<td>6.74</td>
<td>6.52</td>
<td>240.09</td>
<td>0.30</td>
</tr>
</tbody>
</table>
The above example, shows forecasts that overlap. It shows a 48-hour forecast that starts at 2005120300 with 3 hour time interval gap between successive records, and the next is a 12 hour forecast that starts at 2005090712 with 1 hour time interval gap, and the following is a 24 hour forecast that starts at 2005090800 with 3 hour time gap. We assume that some sort of delimiter exists between forecasts.

The format of the timestamp that we use for forecast data is: yyyymmddhh where y - year, m - month, d - day, h - hour.

The parameters of interest are mentioned in bold which include timestamp, temperature, wind speed, wind direction, station pressure.
Appendix B

Observation data is retrieved for the first forecast period mentioned in Appendix A. The program calculates the number of hours required to retrieve the corresponding observation data from the current system time which turns out to be 68 hours when the program was run on 2005-12-05 at 19:50 GMT.

Observation data Format

Observation data for PABR for 68 hours on 2005-12-05 at 19:50 GMT

StnId, Lat, Lon, Elev, Time, Tmp, Dwpt, RH, WndChl, WndDir, WndSpd, Vsby, WX, Clds, SLA, Altim, StnPres, Pcpn 1 hr, Pcpn 3 hr, Pcpn 6 hr, Snow Dpth, Snow 6 hr, Snow 24 hr, 6 hr Max T, 6 hr Min T, 24 hr Max T, 24 hr Min T, QC.

PABR, 71.29, -156.77, 43, 200512030050, -1, -5.82, 50.7, 8.00, -SN, SCT046 OVC065, 1027.9, 30.35, 30.312, T, Caution.

PABR, 71.29, -156.77, 43, 200512030150, 0, -4.83, 40.9, 9.00, -SN, OVC070, 1028.1, 30.36, 30.322, T, Caution.

PABR, 71.29, -156.77, 43, 200512030250, 0, -4.83, 50.8, 5.00, -SN, OVC070, 1028.4, 30.37, 30.332, T, Caution.

PABR, 71.29, -156.77, 43, 200512030320, 0, -4.84, 56.8, 8.00, -SN, BKN015, BKN055, OVC070, 30.37, 30.332, T, Caution.

PABR, 71.29, -156.77, 43, 200512030350, 0, -4.83, 60.7, 7.00, -SN, SCT015, OVC050, 1028.6, 30.37, 30.332, T, Caution.

PABR, 71.29, -156.77, 43, 200512030450, 1, -4.79, 60.5, 9.00, -SN, SCT018, OVC046, 1028.8, 30.38, 30.342, T, Caution.

PABR, 71.29, -156.77, 43, 200512030505, 1, -4.77, 70.5, 9.00, -SN, BKN018, OVC046, 30.38, 30.342, T, Caution.

PABR, 71.29, -156.77, 43, 200512030540, 1, -6.71, 60.3, 7.00, -SN, FEW018, OVC044, 30.38, 30.342, T, Caution.

PABR, 71.29, -156.77, 43, 200512030550, 1, -5.75, 60.3, 7.00, -SN, FEW020, OVC044, 1029.1, 30.39, 30.352, T, T, 7.00, ..., 1., Caution.

PABR, 71.29, -156.77, 43, 200512030605, 1, -4.77, 70.3, 7.00, -SN, BKN020, OVC044, 30.39, 30.352, T, Caution.

PABR, 71.29, -156.77, 43, 200512030650, 0, -4.83, 0.9, 9.00, -SN, BKN020, BKN025, OVC045, 1029.4, 30.40, 30.362, T, Caution.

PABR, 71.29, -156.77, 43, 200512030730, 1, -4.77, 0.9, 9.00, SCT022, OVC043, 30.40, 30.362, T, Caution.

PABR, 71.29, -156.77, 43, 200512030740, 1, -4.77, 0.9, 8.00, -SN, BKN020, OVC043, 30.40, 30.362, T, Caution.

PABR, 71.29, -156.77, 43, 200512030750, 1, -4.79, 0.7, 7.00, -SN, BKN020, OVC043, 1029.8, 30.41, 30.372, T, Caution.

PABR, 71.29, -156.77, 43, 200512030850, 1, -5.82, -16.50, 9.9, 9.00, -SN, BKN018, 1030.3, 30.42, 30.382, T, ..., 2., Caution.

PABR, 71.29, -156.77, 43, 200512030900, -2, -6.84, 18.40, 9.9, 9.00, SCT018, 30.42, 30.382, T, Caution.

PABR, 71.29, -156.77, 43, 200512030950, -1, -5.82, 19.50, 12.9, 9.00, BKN014, 1030.7, 30.43, 30.392, T, Caution.

PABR, 71.29, -156.77, 43, 200512031025, -2, 8.77, -18.50, 9.10, 0.00, FEW014, 30.43, 30.392, T, Caution.

PABR, 71.29, -156.77, 43, 200512031050, -4, -8.84, 20.60, 9.10, 0.00, FEW014, 1030.8, 30.44, 30.402, T, Caution.

PABR, 71.29, -156.77, 43, 200512031055, -4, -8.84, 20.60, 9.10, 0.00, FEW014, 30.44, 30.402, Caution.

PABR, 71.29, -156.77, 43, 200512031150, -4, -8.84, 70.8, 10.00, OVC070, 1031.1, 30.44, 30.402, T, ..., 2., Caution.
## Appendix C

### index.html

### Weather Analysis...

<table>
<thead>
<tr>
<th>Date</th>
<th>Location</th>
<th>Hours</th>
<th>Start Time</th>
<th>Interval Gap</th>
</tr>
</thead>
<tbody>
<tr>
<td>2005-12-03_03</td>
<td>PABR</td>
<td>48.0</td>
<td>03</td>
<td>3.0</td>
</tr>
<tr>
<td>2005-11-18_13</td>
<td>PABR</td>
<td>12.0</td>
<td>13</td>
<td>1.0</td>
</tr>
<tr>
<td>2005-11-17_03</td>
<td>PABR</td>
<td>24.0</td>
<td>03</td>
<td>3.0</td>
</tr>
</tbody>
</table>
### Appendix D

#### Html file Output

<table>
<thead>
<tr>
<th>Time</th>
<th>Fore_Temp</th>
<th>Obs_Temp</th>
<th>Diff_TempF</th>
<th>Diff_TempF</th>
<th>Fore_Pras</th>
<th>Obs_Pras</th>
<th>Diff_Pras</th>
<th>Diff_Pras</th>
<th>Fore_WindSpd</th>
<th>Obs_WindSpd</th>
<th>Diff_WindSpdF</th>
<th>Fore_WindDir</th>
<th>Obs_WindDir</th>
<th>Diff_WindDirF</th>
</tr>
</thead>
<tbody>
<tr>
<td>2005-12-03_03</td>
<td>13.79</td>
<td>-17.78</td>
<td>31.57</td>
<td>927.91</td>
<td>1027.15</td>
<td>-99.24</td>
<td>3.35</td>
<td>6.372</td>
<td>-3.022</td>
<td>210.16</td>
<td>50.0</td>
<td>160.16</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2005-12-03_06</td>
<td>10.88</td>
<td>-17.22</td>
<td>27.9</td>
<td>926.54</td>
<td>1027.83</td>
<td>-101.29</td>
<td>4.23</td>
<td>2.606</td>
<td>-1.624</td>
<td>192.95</td>
<td>66.66</td>
<td>126.29</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2005-12-03_09</td>
<td>10.74</td>
<td>-18.89</td>
<td>29.63</td>
<td>927.28</td>
<td>1028.85</td>
<td>-101.57</td>
<td>1.52</td>
<td>7.820</td>
<td>-6.3</td>
<td>163.35</td>
<td>40.0</td>
<td>123.35</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2005-12-03_12</td>
<td>9.02</td>
<td>-19.44</td>
<td>28.46</td>
<td>927.22</td>
<td>1029.60</td>
<td>-102.47</td>
<td>4.67</td>
<td>6.517</td>
<td>-1.847</td>
<td>121.22</td>
<td>70.0</td>
<td>51.22</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2005-12-03_15</td>
<td>8.26</td>
<td>-17.78</td>
<td>26.04</td>
<td>928.54</td>
<td>1029.94</td>
<td>-103.4</td>
<td>9.24</td>
<td>6.372</td>
<td>2.808</td>
<td>125.45</td>
<td>84.44</td>
<td>41.01</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2005-12-03_18</td>
<td>9.85</td>
<td>-17.78</td>
<td>27.63</td>
<td>925.05</td>
<td>1030.20</td>
<td>-105.15</td>
<td>10.30</td>
<td>0.617</td>
<td>9.648</td>
<td>134.04</td>
<td>62.5</td>
<td>71.54</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2005-12-03_21</td>
<td>11.21</td>
<td>-16.11</td>
<td>27.32</td>
<td>924.48</td>
<td>1030.28</td>
<td>-105.8</td>
<td>15.36</td>
<td>11.10</td>
<td>4.26</td>
<td>152.63</td>
<td>82.22</td>
<td>70.41</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2005-12-04_00</td>
<td>10.49</td>
<td>-18.89</td>
<td>29.38</td>
<td>924.20</td>
<td>1031.57</td>
<td>-107.37</td>
<td>10.88</td>
<td>10.52</td>
<td>0.36</td>
<td>145.11</td>
<td>80.36</td>
<td>64.75</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2005-12-04_03</td>
<td>11.11</td>
<td>-22.22</td>
<td>33.33</td>
<td>923.67</td>
<td>1031.95</td>
<td>-108.28</td>
<td>10.55</td>
<td>5.069</td>
<td>5.681</td>
<td>136.46</td>
<td>70.33</td>
<td>58.13</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2005-12-04_06</td>
<td>8.39</td>
<td>-23.33</td>
<td>31.72</td>
<td>921.75</td>
<td>1032.08</td>
<td>-110.93</td>
<td>7.69</td>
<td>7.965</td>
<td>-0.275</td>
<td>123.89</td>
<td>90.0</td>
<td>33.89</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2005-12-04_09</td>
<td>8.52</td>
<td>-23.33</td>
<td>31.65</td>
<td>923.31</td>
<td>1033.59</td>
<td>-110.28</td>
<td>3.98</td>
<td>0.0</td>
<td>3.08</td>
<td>227.38</td>
<td>0.0</td>
<td>227.38</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2005-12-04_12</td>
<td>7.91</td>
<td>-24.44</td>
<td>32.35</td>
<td>925.86</td>
<td>1033.93</td>
<td>-108.07</td>
<td>6.82</td>
<td>3.910</td>
<td>2.91</td>
<td>217.29</td>
<td>120.0</td>
<td>97.29</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2005-12-04_15</td>
<td>6.85</td>
<td>-21.11</td>
<td>27.96</td>
<td>926.60</td>
<td>1034.27</td>
<td>-107.67</td>
<td>6.53</td>
<td>5.069</td>
<td>1.461</td>
<td>213.58</td>
<td>211.6</td>
<td>1.98</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2005-12-04_18</td>
<td>5.94</td>
<td>-20.0</td>
<td>25.94</td>
<td>927.14</td>
<td>1034.32</td>
<td>-107.18</td>
<td>7.31</td>
<td>5.358</td>
<td>1.952</td>
<td>203.70</td>
<td>225.0</td>
<td>-21.3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2005-12-04_21</td>
<td>6.16</td>
<td>-20.0</td>
<td>26.16</td>
<td>929.42</td>
<td>1034.66</td>
<td>-105.24</td>
<td>7.49</td>
<td>5.538</td>
<td>1.552</td>
<td>216.85</td>
<td>240.0</td>
<td>-23.15</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2005-12-05_00</td>
<td>6.88</td>
<td>-18.89</td>
<td>25.77</td>
<td>931.64</td>
<td>1034.94</td>
<td>-103.3</td>
<td>6.27</td>
<td>7.820</td>
<td>-1.55</td>
<td>243.58</td>
<td>219.2</td>
<td>24.38</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Appendix F

Flow Chart

start

retry after some time

status code 200 / 404 / 500

get observational data

status code 200 with data / 301

no

yes

store in file

synchronize

compare

plot graph for Observational and Forecast Data

retry

upload data

no

successful

yes

end

get forecast data

no

successful

yes

store in file
Appendix G

Program Flow

```
./runme

Main_Read.py

Main_Read.Calculate_Hours() GetObservationData.Create_URL() FileEstimate.FileOpr()

SendRequest.Fetch() Compare.compare()

ErrorHandler.py

Generate_Summary.Upload_File() Generate_HTML.Create_HTML() PlotGraph.plot()

Rotate.Angle()
```
Description

Initially, the program is run with ./runme providing arguments. It then sends these user input arguments to the Main_Read.py file. Main_Read.py processes these input arguments and reports the user if anyone of the required arguments are missing and then exits. If all the arguments are fine, it then reads the first forecast period from the forecast file and then calls the Main_Read.Calculate_Hours() function, to calculate the number of Hours required to retrieve Observation data.

Once it gets the number of hours, it then calls GetObservationData.Create_URL() to create a url request object. This function then calls the SendRequest.Fetch() to actually retrieve the observation data from the NWS website, passing the required parameters (station id and number of hours). Fetch() may call functions defined in ErrorHandler.py if any exceptions raise during the retrieval of observed data. When all the required observational data is retrieved, the program control goes back to Main_Read.py. It then calls the FileEstimate.FileOpr() passing observational file object and the forecast period data as arguments, to synchronize observational and forecast formats.

FileOpr() synchronizes observational data with forecast data and then calls Compare.compare() to calculate the difference in observation and forecast values. It then either plots graphs, or generates text file based on user’s input.

If the user selects text output, it calls Generate_Summary.Upload_File() to upload the text file onto the server using public encryption.

If the user selects web output, it then calls PlotGraph.plot() which plots the graphs. PlotGraph.plot() calls the Rotate.Angle() to rotate the generated graph by 270 degrees. Once the graphs are plotted by PlotGraph.plot, the program control goes back to
Compare.compare( ) then calls Generate_Html.Create_HTML( ) to generate a html file with a table of data and graphs. Once the HTML file is generated it uploads the files onto the remote server using public encryption. Once all this is done, the program control goes back to Compare.compare( ) and then back to FileEstimate.FileOpr( ) and back to Main_Read.py.

Main_Read.py repeats this process for each forecast period in the forecast file.

For all the above operations, the scripts use the constants defined in Const_Variables.py file.
Appendix H

Python Scripts

Refer to accompanying cd.
Appendix I

Installing Software's

The following software's should be installed, in the following order, for running my scripts.

- Python-2.4.2 should be installed first.
- Gnuplot-4.0.0, Numeric-24.0 packages should be installed before gnuplot-1.7 (interface module between Python-2.4.2 and gnuplot-4.0.0) can be installed.
- Python should be installed before PIL (Python Imaging Library) or HTMLgen is installed.

Python is already installed under RedHat (as /usr/bin/python, with associated files in /usr/lib/). I prefer to install a separate version, so that I can get access to latest versions. Make sure that the new version that you install is in your path ahead of the default version in /usr/bin.

Installing Python

- Download the latest version of the software at Python download site, www.python.org/download. The version that I used for my scripts is Python-2.4.2. To download, click on "Python 2.4.2 Source (for Unix or OS X compile)". This will download a gzipped tar file containing the Python distribution.
- Untar the tarball: tar xvfz Python-2.3.4.tgz
  
  This will create a directory named with the appropriate version number, e.g., Python-2.4.2. cd into that directory:
cd Python-2.4.2

- You will need to decide where you are going to install the software once it's built. This place is generally known as the PREFIX (in configure scripts). Python will install files in several places under PREFIX, e.g., PREFIX/bin, PREFIX/lib, PREFIX/include. If you have root permission on your system, /usr/local is generally a good place to do it (and, in fact, /usr/local is the default place Python will try to install itself into if you do not specify a different location). If you do not have root permission, you can install within your own filesystem, e.g., /home/username.

- Python gets built through a standard configure-make-make install process. First run configure, specifying the installation location (or leaving it out if you want to install in /usr/local).

  ./configure (to install in /usr/local) or
  ./configure --PREFIX=/home/username (to install in /home/username)

- Once configure is finished, run "make" to build the software.

  make

- Once make is finished, run "make install" to install the software.

  make install

- Assuming this has all worked correctly, the Python interpreter has been installed as PREFIX/bin/python (e.g., /usr/local/bin/python), and the Python standard library has been installed under PREFIX/lib/python2.4. Rearrange your path (e.g., by editing your .bashrc or .cshrc) to ensure that the newly installed Python is found prior to the default system Python.
As we install additional Python-related software which is not part of the core Python distribution, these will be placed in the site-packages directory under PREFIX/lib/python2.4, e.g., in /usr/local/lib/python2.4/site-packages. When you call the Python interpreter, it knows to look in the site-packages directory for modules that are imported.

Information on building and installing Python is contained in the README file in Python-2.4.2 directory, for more information.

Installing gnuplot-1.7

Before you can use Gnuplot.py, you will need working versions of

- the gnuplot program,
- Python (version 1.5 or later), and
- the Numeric Python extension

To install Gnuplot.py:

- Download either gnuplot-py-1.7.tar.gz or Gnuplot-1.7.zip.
- Gunzip and untar (or unzip) it, which will create a directory called gnuplot-1.7.
- Refer to README.txt in that directory for further instructions if rpm needs to be installed. Usually it should be enough to type python setup.py install.

Installing Gnuplot-4.0.0
Download gnuplot-4.0.0 from www.pythonware.com

- cd to the gnuplot-4.0.0 directory
  
  cd gnuplot-4.0.0

- Run configure, specifying the installation location
  
  ./configure

- Once configure is finished, run "make" to build the software.
  
  make

- Once make is finished, run "make install" to install the software.
  
  make install

- Then run setup.py
  
  cd ..

  python setup.py install

Installing Numerical Python (NumPy, or Numeric)

NumPy is a Python extension module providing support for efficient manipulation of arrays and matrices, and associated linear algebra functions.


- Untar the Numeric tarball in a directory where you have write permission (e.g., the same place you installed Python), as in

  cd /usr/local

  tar xvfz Numeric-23.3.tar.gz

- cd to the directory created by the untarring process, e.g.,
  
  cd Numeric-23.3
Then run the command

    python setup.py install

**Installing Python Imaging Library (PIL)**

Download Imaging-1.1.5 from www.pythonware.com

- cd to the libImaging directory

    cd libImaging

- Run configure, specifying the installation location

    ./configure

- Once configure is finished, run "make" to build the software.

    make

- Once make is finished, run "make install" to install the software.

    make install

- Then run setup.py

    cd ..
    python setup.py install

**Installing HTMLgen**


- Unpack the tar file. This will create a directory called HTMLgen.

- If you have GNU make you can run the test and install targets of

    the provided Makefile. Like,

    cd HTMLgen

    gmake test

    gmake install