Practice Project: Historical Weather Forecast Comparison to Actuals

Estimated time needed: **30** minutes

Learning objectives

In this practice project, you will:

- Initialize your log file
- Write a Bash script to download, extract, and load raw data into a report
- Add some basic analytics to your report
- Schedule your report to update daily
- Measure and report on historical forecasting accuracy

We've broken this project down into manageable steps. Feel free to try any or all of these on your own; however, we recommend checking your work with the details provided.

Exercise 1 - Initialize your weather report log file

1.1 Create a text file called `rx_poc.log`

`rx_poc.log` will be your POC weather report log file, or a text file which contains a growing history of the daily weather data you will scrape. Each entry in the log file corresponds to a row as in **Table 1**.

▶ Click here for Hint
▶ Click here for Solution

1.2 Add a header to your weather report

Your header should consist of the column names from **Table 1**, delimited by tabs.
Write the header to your weather report.

▶ Click here for Hint
▶ Click here for Solution

> **Tip**: Although it might seem redundant, it is better practice to use variables in these cases. Variables make for much cleaner code, which is easier to understand and safer to modify by others or even yourself at a later date. Using meaningful names for your variables also makes the code more "self-documenting."

Exercise 2 - Download the raw weather data

2.1. Create a text file called `rx_poc.sh` and make it an executable Bash script

▶ Click here for Hint 1
▶ Click here for Hint 2
▶ Click here for Solution 1
▶ Click here for Solution 2

2.2. Assign the city name to `Casablanca` for accessing the weather report

▶ Click here for Hint
▶ Click here for Solution

2.3 Obtain the weather information for Casablanca

▶ Click here for Hint
▶ Click here for Solution

Exercise 3 - Extract and load the required data

3.1. Edit `rx_poc.sh` to extract the required data from the raw data file and assign them to variables `obs_temp` and `fc_temp`

Extracting the required data is a process that will take some trial and error until you get it right. Study the weather report you obtained in Step 2.3, determine what you need to extract, and look for patterns.

You are looking for ways to 'chip away' at the weather report by:

- Using shell commands to extract only the data you need (the **signal**)
- Filtering everything else out (the **noise**)
- Combining your filters into a pipeline (recall the use of **pipes** to chain **filters** together)

▶ Click here for a Hint to get started

3.1.1. Extract the current temperature, and store it in a shell variable called `obs_temp`

Remember to validate your results.

You may have noticed by now that the temperature values extracted from *wttr.in* are surrounded by special formatting characters. These "hidden" characters cause the numbers to display in specific color - for example, when you use the `cat` command to display your log file.

Unfortunately you cannot perform arithmetic calculations on such formatted text, so you will need to extract the values from the surrounding formatting so you can make use of them later in this lab.

▶ Click here for Hint 1
▶ Click here for Hint 2
▶ Click here for Solution

3.1.2. Extract tomorrow's temperature forecast for noon, and store it in a shell variable called `fc_temp`

▶ Click here for Hint
▶ Click here for Solution

3.2. Store the current day, month, and year in corresponding shell variables

▶ Click here for Hint
▶ Click here for Solution

3.3. Merge the fields into a tab-delimited record, corresponding to a single row in Table 1

Append the resulting record as a row of data to your weather log file.

▶ Click here for Hint
▶ Click here for Solution

Exercise 4 - Schedule your Bash script `rx_poc.sh` to run every day at noon local time

4.1. Determine what time of day to run your script

Recall that you want to load the weather data coresponding to noon, local time, in Casablanca every day.

First, check the time difference between your system's default time zone and UTC.

▶ Click here for Hint 1
▶ Click here for Solution

4.2 Create a cron job that runs your script

▶ Click here for Hint
▶ Click here for Solution

4.3 Full solution

For reference, here is a Bash script that creates the raw weather report. Try to follow all the steps on your own before looking!

▶ Click here for Full Solution

Exercise 5 - Create a script to report historical forecasting accuracy

Now that you've created an ETL shell script to gather weather data into a report, let's create another script to measure and report the accuracy of the forecasted temperatures against the actuals.

- To start, create a tab-delimited file named `historical_fc_accuracy.tsv`.

Insert the following code into the file to include a header with column names:

```
echo -e "year\tmonth\tday\tobs_temp\tfc_temp\taccuracy\taccuracy_range" > historical_fc_accuracy.tsv
```

One key difference between this report and the previous report you generated is that the forecast temperature will now be aligned with the date the forecast is for. As a result, the date will be in the same row as the observed temperature for that date, rather than the previous row on the day that the forecast was made.

- Also create an executable Bash script called `fc_accuracy.sh`.

Rather than scheduling your new script to run periodically, think of it as a tool you can use to generate the historical forecast accuracy on demand.

5.1. Determine the difference between today's forecasted and actual temperatures

Rather than writing the script to process all of the data at once, let's simplify by solving the problem for just one instance. Later you can modify the script to handle the general case of mupltiple days.

5.1.1. Extract the forecasted and observed temperatures for today and store them in variables

▶ Click here for Hint 1
▶ Click here for Hint 2
▶ Click here for Solution

5.1.2. Calculate the forecast accuracy

▶ Click here for Hint
▶ Click here for Solution

> **Tip**: Your weather report must have at least two days of data for this calulation to make sense.
> To test your code you can simply append some artificial data to your weather report, `rx_poc.log`.

5.2. Assign a label to each forecast based on its accuracy

Let's set the accuracy labels according to the range that the accuracy fits most tightly within, according to the following table. Validate your result.

accuracy range	accuracy label
+/- 1 deg	excellent
+/- 2 deg	good
+/- 3 deg	fair
+/- 4 deg	poor

▶ Click here for Hint 1
▶ Click here for Solution

5.3. Append a record to your historical forecast accuracy file.

▶ Click here for Hint
▶ Click here for Solution

5.4. Full solution for handling a single day

Below is the final script of `fc_accuracy.sh` for handling the accuracy calculations based on just one instance, or day.

▶ Click here for Solution

5.5. Generalization to all days

We leave it as an exercise for you to generalize your code to create the entire weather accuracy history. In the next exercise, you will download and work with a synthetic version of this weather accuracy history report.

Here are some suggestions to guide you should you wish to create the weather accuracy history yourself:

- Iterate through your weather log file using a for loop. On each iteration:
  - Use `head` and `tail` to extract consecutive pairs of lines on each iteration
    - This provides you with the current and previous day's data
  - Treat this pair of lines like you did in your code as yesterday and today's data
  - Perform your accuracy calulations as before
  - Use the correct row to extract date information
  - Append your resulting data to your historical forecast accuracy report

Exercise 6 - Create a script to report weekly statistics of historical forecasting accuracy

In this exercise, you will download a synthetic historical forecasting accuracy report and calculate some basic statistics based on the latest week of data.

Begin by creating an executable bash script called `weekly_stats.sh`.

6.1. Download the synthetic historical forecasting accuracy dataset

Run the following command in the terminal to download the dataset to your current working directory.

```
wget https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMSkillsNetwork-LX0117EN-Coursera/labs/synthetic_historical
```

6.2. Load the historical accuracies into an array covering the last week of data

Remember to make your script executable. Also validate your result by printing the array to the terminal.

▶ Click here for Hint1
▶ Click here for Hint2
▶ Click here for Solution

6.3. Display the minimum and maximum absolute forecasting errors for the week

Now use your array to calculate the minimum and maximum absolute errors over the last week. For example, if you have a vaule of -1, change it to be 1. Echo the minimum and maximum absolute errors to the terminal.

▶ Click here for Hint1
▶ Click here for Hint2
▶ Click here for Solution

Summary

Congratulations! You've just completed a challenging, real-world practice project using many of the concepts you've learned from this course. The knowledge you've gained has prepared you to solve many practical real world problems. You're almost finished with this course now, and the final step in your journey is to complete the peer-reviewed Final Project.

In this lab, you learned how to:

- Initialize your weather report log file
- Write a Bash script that downloads the raw weather data, and extracts and loads the required data
- Schedule your Bash script `rx_poc.sh` to run every day at noon local time
- Apply advanced Bash scripting to produce reporting metrics
- Create a script to report historical forecasting accuracy
- Create a script to report the minimum and maximum absolute errors for the week

Authors

Jeff Grossman

Other Contributors

Rav Ahuja