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

Use the `touch` command or open a new text file from the GUI.

▼ Click here for Solution

Create `rx_poc.log` by entering the following in your terminal:

```
1
```

```
touch rx_poc.log
```

Copied!

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

Use the `echo` command with the `-e` option, and include tab separators `\t` in a string of names.
Consider why the `-e` option is needed.

▼ Click here for Solution

Use a shell variable and command substitution:

```
1
2
```

```
header=$(echo -e "year\tmonth\tday\tobs_tmp\tfc_temp")
echo $header>rx_poc.log
```

Copied!

OR more directly, use `echo` and redirection:

```
1
```

```
echo -e "year\tmonth\tday\thour\tobs_tmp\tfc_temp">rx_poc.log
```

Copied!

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

Use the `chmod` command.

▼ Click here for Solution 1

Create the file `rx_poc.sh`

```
1
```

```
touch rx_poc.sh
```

Copied!

Include the Bash shebang on the first line of `rx_poc.sh`:

```
1
```

```
#! /bin/bash
```

Copied!

▼ Click here for Solution 2

Make your script executable by running the following in the terminal:

```
1
```

```
chmod u+x rx_poc.sh
```

Copied!

Verify your changes using the `ls` command with the `-l` option.

2.2 Edit `rx_poc.sh` to download today's weather report from *wttr.in*

It's good practice to keep a copy of the raw data you are using to extract the data you need.

By appending a date or time stamp to the file name, you can ensure its name is unique. This builds a history of the weather forecasts which you can revisit at any time to recover from errors or expand the scope of your reports.

Using the prescribed date format ensures that when you sort the files, they will be sorted chronologically. It also enables searching for the report for any given date.

If needed, you can compress and archive the files periodically. You can even automate this process by scheduling it.

Follow the steps below to download and save your report as a datestamped file named `raw_data_<YYYYMMDD>`, where `<YYYYMMDD>` is today's date in Year, Month, and Day format.

2.2.1 Create the filename for today's *wttr.in* weather report

▼ Click here for Hint 1

Use the `date` command with the correct formatting option in `rx_poc.sh`.

▼ Click here for Hint 2

Use command substitution to save the date to a variable called `today`.

▼ Click here for Hint 3

Include the variable's value in the name of the file.

▼ Click here for Solution to Hints 1 and 2

Edit `rx_poc.sh` to include:

```
1
```

```
today=$(date +%Y%m%d)
```

Copied!

▼ Click here for full Solution

Edit `rx_poc.sh` to include:

```
1
2
```

```
today=$(date +%Y%m%d)
weather_report=raw_data_$today
```

Copied!

OR more directly:

```
1
```

```
weather_report=raw_data_$(date +%Y%m%d)
```

Copied!

2.2.2 Download the *wttr.in* weather report for Casablanca and save it with the filename you created

▼ Click here for Hint

Use the `curl` command with the `--output` option.

▼ Click here for Solution

Edit `rx_poc.sh` to include:

```
1
2
```

```
city=Casablanca
curl wttr.in/$city --output $weather_report
```

Copied!

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_tmp` 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 downloaded, 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

Extract only those lines that contain temperatures from the weather report, and write your result to file.

▼ Click here for another Hint to get started

Use `grep` and redirect the result to file.

▼ Click here for Solution

Edit `rx_poc.sh` to include:

```
1
```

```
grep °C $weather_report > temperatures.txt
```

Copied!

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

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 characters cause the numbers to display in specific color - for example, when you use the `cat` command to display your log file.

To see these "hidden" characters, you can use an editor such as Vim. When you use Vim or another editor, your records will look something like this:

```
2023 05 16 ^[[38;5;190m20^[[0m ^[[38;5;190m21^[[0m
```

Tip: You can use `cat` with the `-A` option to display all hidden characters.

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. For this example record, you need to strip the values `20` and `21` from the string.

▶ Click here for Hint 1

▼ Click here for Hint 2

Use the `cat` command with the correct option to print the all characters, including hidden ones, to a pipeline.

▼ Click here for Hint 3

Are there any characters you can use as a delimiter to appropriately parse the line into fields?

▼ Click here for Solution

While adding the following lines to `rx_poc.sh`, ensure you understand what each filter accomplishes by using the command line. Try adding one filter at a time to see what the outcome is as you develop the pipeline.

```
1
2
3
```

```
# extract the current temperature
obs_tmp=$(cat -A temperatures.txt | head -1 | cut -d "+" -f2 | cut -d "m" -f5 | cut -d "^" -f1 )
echo "observed temperature = $obs_tmp"
```

Copied!

Note that sometimes the wttr report precedes temperatures with a plus symbol `+`, but not necessarily. There is no harm in including the `cut -d "+" -f2` filter because the text will pass through it unchanged.

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

▼ Click here for Hint

Provided you understand the previous pipeline, you will be able to solve this problem through experimentation.

▼ Click here for Solution

Add to `rx_poc.sh`:

```
1
2
```

```
# extract the forecast for noon tomorrow
fc_temp=$(cat -A temperatures.txt | head -3 | tail -1 | cut -d "+" -f2 | cut -d "(" -f1 | cut -d "^" -f1 )
```

Copied!

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

▼ Click here for Hint

Use command substitution and the `date` command with the correct formatting options.
The time zone for Casablanca is UTC+1. To get the local time for Casablanca, you can set the time-zone environment variable, `TZ`, as follows:

```
1
```

```
TZ='Morocco/Casablanca'
```

Copied!

▼ Click here for Solution

Add to `rx_poc.sh`:

```
1
2
3
4
```

```
hour=$(TZ='Morocco/Casablanca' date -u +%H)
day=$(TZ='Morocco/Casablanca' date -u +%d)
month=$(TZ='Morocco/Casablanca' date +%m)
year=$(TZ='Morocco/Casablanca' date +%Y)
```

Copied!

Note: You might be wondering why we didn't just set `hour` to a value of 12, since we want to get the time at noon.
However, if we did, we would lose the ability to verify that we are actually running the code at the correct *local* time.
You should think of the local time as a *measurement* rather than a set number.

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

How did you create the header to initialize your log file?

▼ Click here for Solution

Add to `rx_poc.sh`:

```
1
2
```

```
record=$(echo -e "$year\t$month\t$day\t$obs_tmp\t$fc_temp")
echo $record>>rx_poc.log
```

Copied!

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

Use the `date` command twice with appropriate options; once to get your system time, and once to get UTC.

▼ Click here for Solution

Running the following commands gives you the info you need to get the time difference between your system and UTC.

```
1
2
3
4
```

```
$ date
Mon Feb 13 11:28:12 EST 2023
$ date -u
Mon Feb 13 16:28:16 UTC 2023
```

Copied!

Now you can determine how many hours difference there are between your system's local time and that of Casablanca. In the example above, we see that the system time relative to UTC is UTC+5 (i.e. 16 - 11 = 5).

We know Casablanca is UTC+1, so the system time relative to Casablanca is 4 hours earlier. Thus to run your script at noon Casablanca time, you need to run it at 8 am.

4.2 Create a cron job that runs your script

▼ Click here for Hint

Edit the crontab file and review the crontab syntax description included in the file.

▼ Click here for Solution

Edit the crontab file:

```
1. 1
```

```
1. crontab -e
```

Copied!

Add the following line at the end of the file:

```
1. 1
```

```
1. 0 8 * * * /home/project/rx_poc.sh
```

Copied!

Save the file and quit the editor.

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

```
 1. 1
 2. 2
 3. 3
 4. 4
 5. 5
 6. 6
 7. 7
 8. 8
 9. 9
10. 10
11. 11
12. 12
13. 13
14. 14
15. 15
16. 16
17. 17
18. 18
19. 19
20. 20
21. 21
22. 22
23. 23
24. 24
25. 25
26. 26
27. 27
28. 28
29. 29
30. 30
31. 31
32. 32
33. 33
34. 34
35. 35
36. 36
37. 37
```

```
 1. #! /bin/bash
 2.
 3. # create a datestamped filename for the raw wttr data:
 4. today=$(date +%Y%m%d)
 5. weather_report=raw_data_$today
 6.
 7. # download today's weather report from wttr.in:
 8. city=Casablanca
 9. curl wttr.in/$city --output $weather_report
10.
11. # use command substitution to store the current day, month, and year in corresponding shell variables:
12. hour=$(TZ='Morocco/Casablanca' date -u +%H)
13. day=$(TZ='Morocco/Casablanca' date -u +%d)
14. month=$(TZ='Morocco/Casablanca' date +%m)
15. year=$(TZ='Morocco/Casablanca' date +%Y)
16.
17. # extract all lines containing temperatures from the weather report and write to file
18. grep °C $weather_report > temperatures.txt
19.
20. # extract the current temperature
21.
```

```
obs_tmp=$(cat -A temperatures.txt | head -1 | cut -d "+" -f2 | cut -d "m" -f5 | cut -d "^" -f1 )

echo "observed = $obs_tmp"

# extract the forecast for noon tomorrow
fc_temp=$(cat -A temperatures.txt | head -3 | tail -1 | cut -d "+" -f2 | cut -d "(" -f1 | cut -d "^" -f1 )
echo "fc temp = $fc_temp"

# create a tab-delimited record
# recall the header was created as follows:
# header=$(echo -e "year\tmonth\tday\tobs_tmp\tfc_temp")
# echo $header>rx_poc.log

record=$(echo -e "$year\t$month\t$day\t$obs_tmp\t$fc_temp")
# append the record to rx_poc.log
echo $record>>rx_poc.log
```

Copied!

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 begin, create a tab-delimited file called `historical_fc_accuracy.tsv` using the following code to insert a header of column names:

```
1
```

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

Copied!

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

Look up the record from yesterday.

▼ Click here for Hint 2

Extract the forecast from the appropriate field.

▼ Click here for Solution

```
1
```

```
yesterday_fc=$(tail -2 rx_poc.log | head -1 | cut -d " " -f5)
```

Copied!

5.1.2. Calculate the forecast accuracy

▼ Click here for Hint

First extract today's observed temperature. Then calculate the difference between the forecasted and observed temperatures.

▼ Click here for Solution

```
1
2
```

```
today_temp=$(tail -1 rx_poc.log | cut -d " " -f4)
accuracy=$(($yesterday_fc-$today_temp))
```

Copied!

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

Use two conditions to compare the accuracy sizes to each positive and negative integer range, accordingly.

▼ Click here for Solution

```
1
2
3
4
5
6
7
8
9
10
11
12
13
14
```

```
if [ -1 -le $accuracy ] && [ $accuracy -le 1 ]
then
   accuracy_range=excellent
elif [ -2 -le $accuracy ] && [ $accuracy -le 2 ]
then
    accuracy_range=good
elif [ -3 -le $accuracy ] && [ $accuracy -le 3 ]
then
    accuracy_range=fair
else
    accuracy_range=poor
fi

echo "Forecast accuracy is $accuracy"
```

Copied!

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 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.

```
1
```

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

Copied!

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

First extract the last week of data.

▼ Click here for Hint2

Store the values in a temporary file called `scratch.txt`. Write the contents of `scratch.txt` into an array called `week_fc`.

▼ Click here for Solution

Your shell script should look something like the following. There are many ways to accomplish this task, so the solution provided is not unique.

```
1
2
3
4
5
6
7
8
9
10
```

```
#!/bin/bash

echo $(tail -7 synthetic_historical_fc_accuracy.tsv  | cut -f6) > scratch.txt

week_fc=($(echo $(cat scratch.txt)))

# validate result:
for i in {0..6}; do
    echo ${week_fc[$i]}
done
```

Copied!

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

Check for any negative values in your array, and reassign these array entries with their positve counterparts.

▼ Click here for Hint2

Initialize two variables, `minimum` and `maximum`. Loop over the array values and modify these two variables as required.

▼ Click here for Solution

Your final shell script should now look something like the following.

```
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
```

```
#!/bin/bash

echo $(tail -7 synthetic_historical_fc_accuracy.tsv  | cut -f6) > scratch.txt

week_fc=($(echo $(cat scratch.txt)))

# validate result:
for i in {0..6}; do
    echo ${week_fc[$i]}
done

for i in {0..6}; do
  if [[ ${week_fc[$i]} < 0 ]]
  then
    week_fc[$i]=$(((-1)*week_fc[$i]))
  fi
  # validate result:
  echo ${week_fc[$i]}
done

```

```
minimum=${week_fc[1]}
maximum=${week_fc[1]}
for item in ${week_fc[@]}; do
   if [[ $minimum > $item ]]
   then
     minimum=$item
   fi
   if [[ $maximum < $item ]]
   then
     maximum=$item
   fi
done

echo "minimum ebsolute error = $minimum"
echo "maximum absolute error = $maximum"
```

Copied!

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

Change Log

Date (YYYY-MM-DD)	Version	Changed By	Change Description
2023-06-07	2.2	Jeff Grossman	add minor clarifications
2023-05-17	2.1	Nick Yi	ID Review
2023-05-17	2.0	Jeff Grossman	add analytics content, simplify solutions, handle parsing glitches
2023-04-27	1.2	Nick Yi	QA Pass
2023-04-24	1.1	Nick Yi	ID Review
2023-02-14	1.0	Jeff Grossman	Create initial version of the project

Copyright (c) 2023 IBM Corporation. All rights reserved.