Practice Project: Historical Weather Forecast Comparison to Actuals

Estimated time needed: **90** minutes

Practice project exercises

Ok! Let's get down to business!
I've broken down the project into manageable steps for you. Feel free to try any or all of these on your own, but I do recommend reviewing the details I've provided either way. Good luck!

Exercise 1 - Initialize your weather report log file

1.1 Create a text file called `rx_poc.log`.

This will be your POC weather report log file, simply 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, my preference is to use variables in such cases.
> It makes 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 - Write a bash script that downloads the raw weather data, and extracts and loads the required data

2.1. Create a text file called `rx_poc.sh` and make it a bash script.

Remember to make it executable.

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

2.2 Download today's weather report from wttr.in

Tip: 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 it's 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, Day format.

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

▶ Click here for Hint 1
▶ Click here for Hint 2
▶ Click here for Hint 3
▶ Click here for Solution to Hints 1 and 2
▶ Click here for full Solution

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

▶ Click here for Hint
▶ Click here for Solution

2.3. 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, and determine what you need to extract. Look for patterns.
You must find a way to 'chip away' at the weather report:

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

  Tip: We introduce three more simple filters here that you will find very useful for your data extraction strategy.

  1. `tr` - trim repeated characters to a single character.

For example, to remove extra spaces from text:

```
1
```

```
echo "There are    too    many spaces in this    sentence." | tr -s " "
```

Copied!

{: codeblock}

2. `xargs` - can be used to trim leading and trailing spaces from a string.

For example, to remove the spaces from the begining and the end of text:

```
1
```

```
echo " Never start or end a sentence with a space. " | xargs
```

Copied!

3. `rev` - reverse the order of characters on a line of text.

Try entering the following commmand:

```
1
```

```
echo ".sdrawkcab saw ecnetnes sihT" | rev
```

Copied!

{: codeblock}

You will find `rev` to be a useful operation to apply in combination with the `cut` command.
For example, suppose you want to access the last field in a delimited string of fields:

```
1
2
```

```
# print the last field of the string
echo "three two one" | rev | cut -d " " -f 1 | rev
```

Copied!

You will also find `xargs` to be a useful operation to apply in combination with the `cut` command.
For example, suppose you want to access the last word in a sentence as above, but there happens to be an extra space at the end:

```
1
2
3
4
```

```
# Unfortunately, this prints the last field of the string, which is empty:
echo "three two one " | rev | cut -d " " -f 1 | rev
# But if you trim the trailing space first, you get the expected result:
echo "three two one " | xargs | rev | cut -d " " -f 1 | rev
```

Copied!

Let’s now return to extracting the temperature data of interest.

▶ Click here for a Hint to get started
▶ Click here for another Hint to get started
▶ Click here for Solution

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

▶ Click here for Hint 1
▶ Click here for Hint 2
▶ Click here for Hint 3
▶ Click here for Solution

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

▶ Click here for Hint
▶ Click here for Solution

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

▶ Click here for Hint
▶ Click here for Solution

> **Tip**: You might be wondering why we didn't just set the hour to be 12. After all, noon is the time we are expecting.
> But then we would lose the ability to verify that we are actually running the code at the correct *local* time.
> Threrfore, you should think of the local time as a *measurement*.

2.5. 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 3 - Schedule your bash script `rx_poc.sh` to run every day at noon local time

3.1. 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.
Check the time difference between your system's default time zone and UTC.

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

Now you can determine how many hours difference there are between your system's local time and that of Casablanca.

▶ Click here for Solution

3.2 Create a cron job that runs your script

▶ Click here for Hint
▶ Click here for Solution

Full solution

For reference, here is a bash script that creates the required weather report.
Try not to peek until after you have attempted to work through every step!

▶ Click here for Hint

Congratulations

Well done! 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 practcal real world problems! You are almost there, the final step in your journey is to complete the peer-reviewed Final Project.

Authors

Jeff Grossman

Other Contributors

Rav Ahuja