Assignment 1

Wenhao Wu

January 24, 2014

Abstract

In this assignment to evaluate the mean, standard deviation and the median from the files, I used three different methods which are 1) Welford method implemented in R to read blocks, and the shell to extract the ArrDelay values only; 2) Sample the CSV files and compute the mean, SD, median from the sample observations; 3) Load the data into SQLite3 and use SQL to query the mean, SD and median. The code files are pushed to `https://github.com/huragok/STA250/tree/master/Assignment1`.

1 Welford method in R and BASH

Here we use a method to combine the mean and SD of multiple subsets of samples to get the mean and SD of the entire set of samples [1]: for any partition of the sample $\mathcal{X}$ into sets $\mathcal{X}_1$, $\mathcal{X}_2$, denote $\bar{x}_i$, σ_i^2, n_i as the mean, variance and cardinal of $\mathcal{X}_i$, respectively. We have

$$\bar{x} = w_1\bar{x}_1 + w_2\bar{x}_2; \tag{1}$$

$$\sigma^2 = w_1\sigma_1^2 + w_2\sigma_2^2 + w_1w_2\delta^2. \tag{2}$$

in which

$$w_i = \frac{n_i}{n_1 + n_2}; \tag{3}$$

$$\delta = \bar{x}_1 - \bar{x}_2. \tag{4}$$

We assume we are given a bunch of *.csv.bz2 files in which the 15-th column is named ArrDelay that contains integers or "NA". We build the following connection to read a file to the standard iostream:

```
command = sprintf("bunzip2 -c %s | cut -f15 -d',' | grep
    -v ArrDelay | grep -v NA", fileName)
connection = pipe(command, 'r')
```

Then we use scan() to read a block of several lines sequentially. We do not want to use readLines() because to read from the position reached last time we have to go through the readied lines again with the offset option. Then we evaluate

the mean and variance of this block. Note that we also evaluate the histogram (or the so-called "discrete frequency table"). Each time we read a new block, we combine its mean, variance with that of the union of all processed blocks with eqn. 12. We also combine the histograms using the set operation in R. After going through each file, we go to the next file.
I have posted a brief analysis on [2] based on graph theory and the effects of finite register length on FFT presented in section 9.7 of [3]. The basic idea is that instead of merging new blocks to the union of processed blocks, we should merge a few blocks into a larger block, then merge a few of these larger blocks into an even larger block, and so forth until we have merged all records. In such a way we actually construct a tree with minimum depth for a given maximum degree for each node. Some really simple analysis shows that this will result in significantly less numerical error.
Also note we actually can evaluate the mean and the variance using the merged histogram so it makes little sense to use Welford method for our problem.

2 Fast sampling method in c++

Although the Welford method implemented in R and BASH is sounded theoretically, the actual data is more complex so that we need to be able to

- Identify the different column numbers of the arrival delay data according to various possible names "ArrDelay" or "ARR_DELAY";
- Customize the cut method so that it can handle delimiters enclosed in quotes.

We decided to implement this method purely in c++ since speed is critical especially when we are processing the data files line-by-line. In terms of the algorithm, we decide to sample a small portion of the original data, construct the histogram and then evaluate the mean, SD and median with it.

The main design focus is to design the histogram class. Since we do not want to make much assumption on the field of arrival delay and we do not want to waste a lot of memory, we decided to implement the histogram with two list links in which each node contains a unique arrival delay value and its frequency. The two list links contains all non-negative/negative numbers and are increasingly/decreasingly ordered, respectively. Since we expect higher frequency for arrival delays with smaller absolute value, these double linked-list structure will significantly increase the speed of linear indexing to add a new category or increase the frequency of an existing frequency by one.

Unfortunately, since I do not have enough disk space on my virtual machine I cannot run a test on the complete data set. Nevertheless, an initial test on a subset of 11 *.csv files with sampling probability of 0.0001 shows that it evaluated all three values really fast [4] (Note it seems that the clock frequency

on my virtue machine is wrong so the actual time is not the same as the result given by the program).

3 Database method in c++ and SQLite3

If we would like to perform other analysis given the data set, it is desirable to store the needed data into a data set and use SQL to get the result directly. In this implementation, we insert all non-empty, non-NA arrival delay into a table in a SQLite3 database. This table contains only two columns: an indexing column (ID) and the arrival delay column (ARRDELAY). Of course we can build a larger table or several tables containing all the information from the csv files. However, this is unnecessary, time-consuming and again we do not have the disk space. The implementation for this method is very similar to that of the fast sampling method. The main design focus is to insert the data into the database as fast as possible. Instead of inserting one record per SQL command, we use the multi-row insert as in MySQL that is supported by SQLite3 3.7.11+, which significantly increase the inserting speed [5]. Note that compound insert is not advantageous compared with wrapping the multiple inserts into a single transaction, however, it is significantly faster than calling the insert command once at a time.
The SQL commands to compute mean and standard deviation are pretty simple. To compute the median, we use the following command [6]:

```
SELECT AVG(ARRDELAY)
FROM (
        SELECT ARRDELAY
        FROM FLIGHTINFO
        ORDER BY ARRDELAY
        LIMIT 2 - (SELECT COUNT(*) FROM FLIGHTINFO) % 2
              OFFSET (SELECT (COUNT(*) - 1) / 2 FROM
                  FLIGHTINFO)
      );
```

An initial test on a subset of two *.csv files shows that it evaluated all three values really fast [4] once the records are inserted into the database.

References

[1] Wikipdia: Parallel algorithm. `http://en.wikipedia.org/wiki/Algorithms_for_calculating_variance#Parallel_algorithm`. Accessed: 2014-01-24.

[2] Followup discussion on piazza. `https://piazza.com/class/hp3u1qqtpk6485?cid=16`. Accessed: 2014-01-24.

[3] Alan V Oppenheim. *Discrete-Time Signal Processing, 2/E*. Pearson Education India, 2006.

[4] Fast sample method. https://github.com/huragok/STA250/tree/master/Assignment1/ArrDelayFastSample. Accessed: 2014-01-24.

[5] Stackoverflow: Compound insert in sqlite3. http://stackoverflow.com/questions/1609637/is-it-possible-to-insert-multiple-rows-at-a-time-in-an-sqlite-database. Accessed: 2014-01-24.

[6] Stackoverflow: How can i calculate the median of values in sqlite? http://stackoverflow.com/questions/15763965/how-can-i-calculate-the-median-of-values-in-sqlite. Accessed: 2014-01-24.