Statistical Analysis of Trip Advisor Review Data

Jonathan E. Keable

February 25, 2016

Student ID 25616196

1 Scripts

1.1 Counting Reviews

This script is designed to count the number of reviews for each hotel, and then print out these results in descending order. (Line breaks have been added here to improve readability, where the 'ćharacter denotes ignore newline in bash)

```
#!/bin/sh
cd "$1"| grep -c -r --include \*.dat '\<Author\>'| \
xargs -n 1 basename | sort -t: -rn -k2 | \
sed -e 's/\.dat//' -e 's/\:/  /'
```

This one liner does a few different operations, which will each be explained in turn.

```
cd "$1"| grep -c -r --include \*.dat '\<Author\>'|
```

First we change directory to the one specified by the first argument. Now grep is used to count the number of reviews in each file, which can be detected from the number of "<Author>" tags in the data file. the `-c` tells grep to count occurrences, and `-r --include \*.dat` tells grep to do this for each .dat file in the directory.

```
xargs -n 1 basename | sort -t: -rn -k2 |
```

Here, the first command strips the full path from the filename to get it in the form `hotel_000.dat`. The next command sorts the results based on the second column, which is the review count for each file, where `-rn` means reverse order numerical sort, i.e. from the highest to lowest value.

```
sed -e 's/\.dat//' -e 's/\:/  /
```

Finaly this sed command cuts the file extension off the filename, and also replaces the colon seperator with a double space instead. This changes the form from `hotel_000.dat:32` to `hotel_000 32`.

1.2 Average Ratings

This script is designed to extract the average rating for each hotel by calculating the mean of the Overall ratings given by each review. Once this is calculated the sorted results are then printed to the shell. (The script begins on the next page)

```
#!/bin/sh
#The folder to operate in is specified by the first argument
path="${PWD}/$1"
cd $path
# Clears the file used to store results
cp /dev/null means.txt
for file in *.dat;
do
        awk '
        BEGIN{
                # Sets the field seperator to close chevron,
                # and initialises variables
                FS = ">";
                hotel_Ratings[""]=0;
                count = 0;
                total = 0;
                mean = 0;
        }
        {
                # Adds all the overall ratings to an array,
                # and counts how many ratings there are
                if ($1 == "<Overall") {
                        hotel_Ratings[count]+=$2;
                        count++;
                }
        }

        END{
                # Sums the ratings, and calculates the mean
                # by dividing by the number of ratings
                for (i in hotel_Ratings){
                        total += hotel_Ratings[i];
                }
                mean = total/count;

                #prints results to a file for storage
                n=split(FILENAME,a,"/");
                n = split(a[n],b,".");
                print b[1] ":␣" mean >> "means.txt"
        }
        ' "$PWD/$file"
done

#sorts the reuslts and prints to standard out
sort -rn -k2 means.txt
```

The script is mainly written in awk, with bash used to iterate over the files, and sort used to sort the results.

```
#!/bin/sh
#The folder to operate in is specified by the first argument
path="${PWD}/$1"
cd $path
# Clears the file used to store results
cp /dev/null means.txt
for file in *.dat;
do
.
.
.
done
```

As the comments explain, the script first changes its working directory to the one specified by the first argument, and then clears the file used to store results if the script has already been run before in that directory. The next command iterates over all the files in the directory, where the file is passed to the awk script for processing.

The awk part of the script can be broken down into its different blocks. First we will look a the `BEGIN` block.

```
BEGIN{
        # Sets the field seperator to close chevron,
        # and initialises variables
        FS = ">";
        hotel_Ratings[""]=0;
        count = 0;
        total = 0;
        mean = 0;
}
```

Here we set the field seperator to `>` so that awk can process the data which is labeled by tags in the file. Later we can look for `<tagname` in the first ”column” as awk understands it. The `hotel_Ratings[""]` array will store each of the overall ratings in the file for each individual review. The `count` variable will be used as the array index, and to count how many review scores are in the array. The `total` variable will store the sum of all the ratings, and `mean` will store the mean when it is calculated later. The reason we store the overall ratings in an array as opposed to simply adding them to the total as we go along is so that when the code is extended later can use the individual ratings to calculate the variance.

Next we look at the block that runs for each line of the file.

```
{
        # Adds all the overall ratings to an array,
        # and counts how many ratings there are
        if ($1 == "<Overall") {
                hotel_Ratings[count]+=$2;
                count++;
        }
}
```

As explained earlier, we can use the tag with a leading open chevron to find the lines in which the overall ratings are stored for each review. The value is then stored in an array, indexed from 0 by using the `count` variable before it is incremented.

Finally the `END` block is used to calculate the mean based on the ratings extracted from the file.

```
END{
        # Sums the ratings, and calculates the mean
        # by dividing by the number of ratings
        for (i in hotel_Ratings){
                total += hotel_Ratings[i];
        }
        mean = total/count;

                #prints results to a file for storage
        n=split(FILENAME,a,"/");
        n = split(a[n],b,".");
        print b[1] ":␣" mean >> "means.txt"
}
```

First the array containing the ratings is iterated over to obtain the total sum of them. The mean is then equal to the sum of the elements divided by the number of elements. Finally the file's name and its associated is appended to the `means.txt` . In order to get the filename without the path we split it around the '/' character and take the last element of the resulting array. Then to strip off the extension we take the first element of the array generated by splitting around the '.' character (note that in awk this array is indexed from 1).

The result of running this script on each file is an unsorted list of means in the form `hotel_000: 43` in the storage file. To sort them however we will use bash.

```
sort -rn -k2 means.txt
```

This is the same as the command used for sorting by review counts, where `-rn` denotes numerical reverse order and `-k2` means sort based on the second column. The file used is where the means were stored by the awk script.

1.3 Statistical Significance Testing

In this script the data from top 2 rated hotels is compared to see if there is a statistically significant difference between their means. To do this we must calculate the means and standard deviations, and compare the t-score to a critical value based on the degrees of freedom. Once again most of the script uses awk, with a few bash commands also used (Script begins on next page).

(Some line breaks have been added in for readability and so that the code will fit on the page, backslash denoting ignore newline in awk and bash)

```
#!/bin/sh
cd reviews_folder
cp /dev/null stats_results.csv
echo "Filename,Mean,Var,SD,N" >> stats_results.csv
files=("$1" "$2")
for i in ${files[@]}
do
        awk '
        BEGIN{
                FS = ">";
                hotel_Ratings[""]=0;
                diffs_Squared[""]=0;
                count = 0;
                total = 0;
                diffs_Total = 0;
                mean = 0;
                sd = 0;
                var = 0;
        }
        {
                if ($1 == "<Overall") {
                        hotel_Ratings[count]+=$2;
                        count++;
                }
        }

        END{
                for (i in hotel_Ratings){
                        total += hotel_Ratings[i];
                }
                mean = total/count;
                for (i in hotel_Ratings){
                        diffs_Squared[i] = \
                        (hotel_Ratings[i] - mean)^2;
                        diffs_Total += diffs_Squared[i];
                }
                if (count > 1){
                        var = diffs_Total/(count - 1);
                }
                sd = sqrt(var);
                n = split(FILENAME,a,"/");
                n = split(a[n],b,".");
                print  b[1] "," mean "," var "," \
                sd "," count >> "stats_results.csv";
        }'  "$i"
done
```

```
cp /dev/null tResults.txt
awk '
        BEGIN{
                FS = ",";
                t = 0;
                df = 0;
                mean1 = 0;
                mean2 = 0;
                var1 = 0;
                var2 = 0;
                n1 = 0;
                n2 = 0;
                gSD = 0;
        }
        {
                if (NR == 2){
                        mean1 = $2;
                        var1 = $3;
                        n1 = $5;
                }
                if (NR == 3){
                        mean2 = $2;
                        var2 = $3;
                        n2 = $5;
                }
        }
        END{
                df = (n1+n2-2);
                gSD = sqrt( ((n1-1)*var1+(n2-1)*var2)/df );
                t = \
                ((mean1 - mean2)/(gSD*sqrt((1/n1)+(1/n2))));
                print "t: " t;
                print t "," df  >> "tResults.txt";
        }
' stats_results.csv

awk '
        BEGIN{FS = ","}
        {
                if(NR >= 2){
                        print "Mean " $1 ": " $2 ", SD: " $4;
                }
        }
        END{}
' stats_results.csv
```

```
awk '
        BEGIN{
                FS=",";
                # For df = 449, at 5% uncertainty, one tailed
                tCrit=1.648
                t = 0;
        }
        {
                if(NR = 1) t = $1;
        }
        END{
                if(t > tCrit){
                        print "1";
                }
                else{
                        print "0";
                }
        }
' "tResults.txt"
```

As this is an extension of the mean calculation it is only necessary to explain new additions. First let us look at the bash code at the start.

```
#!/bin/sh
cd reviews_folder
cp /dev/null stats_results.csv
echo "Filename,Mean,Var,SD,N" >> stats_results.csv
files=("$1" "$2")
for i in ${files[@]}
do
.
.
.
done
```

Once again the working directory is changed, although this time it is hard coded. This time a comma separated values file is used to store results. Here the file is cleared, and a header line added describing what is stored in each column of the file. The awk script is then applied to each of the files specified in arguments 1 and 2, with results for each being printed to the same results file.

Again we can look at each block of the awk script seperately, starting with the BEGIN block.

```
BEGIN{
        FS = ">";
        hotel_Ratings[""]=0;
        diffs_Squared[""]=0;
        count = 0;
        total = 0;
        diffs_Total = 0;
        mean = 0;
        sd = 0;
        var = 0;
}
```

The variables added here are used to calculate unbiased estimators for the variance and standard deviation, which are stored in `var` and `sd` respectively. The `diffs_Squared[""]` array stores the difference between each overall rating and the mean of all overall ratings, and `diffs_Total` stores their sum.

The way the script reads the overall ratings from the files is the same as before, so we can skip straight to the END block.

```
END{
        for (i in hotel_Ratings){
                total += hotel_Ratings[i];
        }
        mean = total/count;
        for (i in hotel_Ratings){
                diffs_Squared[i] = \
                (hotel_Ratings[i] - mean)^2;
                diffs_Total += diffs_Squared[i];
        }
        if (count > 1){
                var = diffs_Total/(count - 1);
        }
        sd = sqrt(var);
        n = split(FILENAME,a,"/");
        n = split(a[n],b,".");
        print  b[1] "," mean "," var "," \
        sd "," count >> "stats_results.csv";
}'  "$i"
```

The mean calculation is also the same as before, but this time we need to calculate the differences from the mean squared as well. To do this we iterate over the array of overall ratings and calculate this value, adding it to an array, and also adding it to a sum. We could have not used an array here, but in case we needed to look at the individual differences later we still do as the overhead is not very high when comparing two files. If we were iterating over many more files it might be worth just adding to a total, and not using an array to store the values.

To calculate the unbiased estimator of the sample variance we divide this total not by n, but by $n-1$, and the unbiased estimator for the standard deviation is simply the square root of the variance. The same procedure as in the previous scrip is used to extract just the name of the file without path or extension and print it to the results file, and the mean, variance, standard deviation and number of ratings are also appended.

This line of bash simply clears a results file for the t-statistic and degrees of freedom.

```
cp /dev/null tResults.txt
```

The next awk script uses the previous results to calculate the t-score and degrees of freedom. Once again we start by looking at the `BEGIN` block.

```
BEGIN{
        FS = ",";
        t = 0;
        df = 0;
        mean1 = 0;
        mean2 = 0;
        var1 = 0;
        var2 = 0;
        n1 = 0;
        n2 = 0;
        gSD = 0;
}
```

Some of these variables store the means (`mean1` and `mean2`), variances (`var1` and `var2`), and number of results (`n1` and `n2`) for each hotel. An array could have been used instead of separate variables for each file, but for just 2 files this seemed unnecessary. The t-score, degrees of freedom and the grand standard deviation will be stored in the `t`, `df` and `gSD` variables respectively.

The script then reads in the values needed from the first results file.

```
{
        if (NR == 2){
                mean1 = $2;
                var1 = $3;
                n1 = $5;
        }
        if (NR == 3){
                mean2 = $2;
                var2 = $3;
                n2 = $5;
        }
}
```

Here we simply read in the correct columns from line 2 into the variables for the first hotel, and the same columns of from line 3 into the variables for the second hotel.

Finally the end block of the script calculates the degrees of freedom and t-score.

```
END{
        df = (n1+n2-2);
        gSD = sqrt( ((n1-1)*var1+(n2-1)*var2)/df );
        t = \
        ((mean1 - mean2)/(gSD*sqrt((1/n1)+(1/n2))));
        print "t:␣" t;
        print t "," df  >> "tResults.txt";
}
```

For comparing 2 samples of different sizes but with the same variances, the degrees of freedom is equal to $n_1 + n_2 - 2$, where n_1 and n_2 are the number of elements in each of the samples. Then we calculate the grand standard deviation, which is as follows for samples with different sizes but the same variance:

$$g.s.d. = \sqrt{\frac{(n_1 - 1)s_{X_1}^2 + (n_2 - 1)s_{X_2}^2}{n_1 + n_2 - 2}} \tag{1}$$

where $s_{X_1}^2$ an $s_{X_2}^2$ are the unbiased estimators for the variance of sample 1 and sample. We can then use this to calculate the t-statistic using the following equation:

$$t = \frac{\bar{X}_1 + \bar{X}_2}{g.s.d. \cdot \sqrt{\frac{1}{n_1} + \frac{1}{n_2}}} \tag{2}$$

Where $\bar{X}_1$ and $\bar{X}_2$ are the means of each sample. Once the script has done these calculations, the t-statistic is printed to the shell, and is also saved to the results file along with the degrees of freedom.

The penultimate awk script simply prints out some of the first results in the format that we would like.

```
BEGIN{FS = ","}
{
        if(NR >= 2){
                print "Mean␣" $1 ":␣" $2 ",␣SD:␣" $4;
        }
}
END{}
```

This prints out the hotel name, mean and standard deviation for each sample.

The final awk script compares the t-statistic to the critical value for the relevant degrees of freedom.

```
BEGIN{
        FS=",";
        # For df = 449, at 5% uncertainty, one tailed
        tCrit=1.648
        t = 0;
}
{
        if(NR = 1) t = $1;
}
END{
        if(t > tCrit){
                print "1";
        }
        else{
                print "0";
        }
}
```

By using the degrees of freedom we calculated earlier, and comparing this to a critical values table, we calculate the critical value we need to use as 1.648. To get this value we also need to decide if the test is one-tailed or two-tailed. Because we want to know if the mean of sample 1 is significantly *greater* than the sample 2 mean we use one-tailed. If we wanted to know if one mean was significantly *different* from another then we would use a two-tailed test.

The t-statistic is then read from the results file and compared to the critical value. If $t > 1.648$ then there there is a statistically significant difference, and a 1 is printed. if $t \leq 1.648$ then there is not a statistically difference and a 0 is printed instead.

2 Hypothesis Testing

Analysis Data In order to Analyse data we have to decide what our *independent* and *dependent* variables are. In this case it is quite obvious what these are; the hotel is *independent* and the rating is *dependent.* This is because we are compiling a list of hotels and then 'measuring' how good each one is by using customer reviews. The hotel is categorical, and in this case we are also measuring the rating to be categorical, or more precisely ordinal as it is a numerical value (between 1 and 5 inclusive).

The Problem Having collected all this data on user reviews, how can we tell if one hotel is actually better than another, or whether a difference in the average review score is simply down do random chance. We need some measure of statistical significance to ascertain this, for which we use the so called student's t-test. To carry out a test we must first however formulate a null hypothesis and an alternative hypothesis. It makes sense to assume that there is *not* a statistically significant difference between two different means until proven otherwise, and that the alternative would be that the difference would be. This can be represented as follows:

H_0**:** There is **no statistically significant difference** between the user rating of **hotel A** and **hotel B**
H_A**:** The user rating of **Hotel A** is **statistically significantly greater** than the user rating of **Hotel B**

Testing our Hypothesis Having formulated our hypothesis we now need to test it using the t-test. We recall the equations we we used earlier:

$$g.s.d. = \sqrt{\frac{(n_1 - 1)s^2_{X_1} + (n_2 - 1)s^2_{X_2}}{n_1 + n_2 - 2}} \tag{1}$$

$$t = \frac{\bar{X}_1 + \bar{X}_2}{g.s.d. \cdot \sqrt{\frac{1}{n_1} + \frac{1}{n_2}}} \tag{2}$$

By applying these equations using our script we can calculate a t-statistic. In this instance the result was $\simeq 0.026$ as we can see from the output of the script:

```
t: 0.0258672
Mean hotel_188937: 4.77953, SD: 0.759441
Mean hotel_203921: 4.77778, SD: 0.59624
0
```

This is not even close to being greater than the critical value we determined for 449 degrees of freedom, which was 1.648. As a result **we do not invalidate our null hypothesis**, so we cannot say that the difference is statistically significant. This is perhaps not surprising considering how similar the two means are, with only about 0.002 between them, and that even the smaller significant difference is ~ 0.6.

3 Description