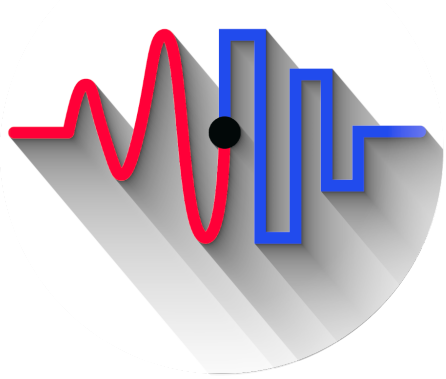

Beats in Bytes

A Statistical Analysis of Melodic Properties

Kai Vernooy, Arin Khare, James Lian

Table of Contents:

Abstract

The goal of Beats in Bytes is to numerically interpret classical music by analyzing it for statistical trends. To achieve this, music files for data collection were collected from online libraries. Code from the Bash programming language was written for data collection. The data we gathered consisted of many attributes per song that measure several aspects of classical melodies, organized into time periods, composers, and songs. Our data revealed strong correlations with previously established distribution laws, such as Zipf's and Benford's laws. New patterns and equations were also uncovered, unlike any other seen before. Additionally, we found that across many different categories, the Renaissance time period is an outlier. We also created a new formula intended to measure the correlation between two different sets of data. The results from applying this formula to the data found that classical music is following the trend of becoming more similar to the Renaissance. Overall, Beats in Bytes found that both common and unusual statistical distributions can be found in music, and that the Renaissance time period is statistically unique, though its style may be showing itself again in future classical pieces. This project was one of a few to utilize code in the topic of classical music analysis, and the methods and tools we developed in this project will be useful to researchers for years to come. Beats in Bytes is the first large-scale case study of the statistical applications of classical music - an art that has influenced humanity for centuries.

Introduction

Beats in Bytes attempts to find the statistical patterns in music. The vast category of 'classical music' can be broken into five main time periods: Renaissance (1400-1610), Baroque (1600-1750), Classical (1730-1820), Romantic (1810-1910), and 20th Century (1901-2000). Inside each of these time periods, we selected five famous composers with 20 random songs each, giving us a total of 500 songs from which we collected our data. From each song, we took certain measurements that quantify an aspect of the melody. We were then able to compare the time periods and composers to each other. Additionally, from averaging the different metrics of all the songs, we were able to create a baseline for each time period, composer, and classical music as a whole. Beats in Bytes found several of the statistical patterns and laws that govern certain styles or time periods in music. We also hypothesized that known distributions laws, such as Zipf's Law or the 80/20 rule (a type of Pareto Distribution) would be found in our data. In addition to these previously discovered laws, we predicted that we would find entirely new curves, equations and distribution rules for many aspects of music. Using all these curves, laws, and comparisons, we were also able to construct a formula that can show how closely two songs, composers, or time periods (or any two of the three) are correlated. This allowed us to find which composers were most similar, and draw conclusions about how closely related various composers and time periods are. Establishing such a formula allows us to compare any melody in music to any other song with ease and as such is extremely powerful. This formula also has applications beyond just classical music, as it could be used to determine correlation for any two sets with the same number of elements. From the outputs of the formula, we were then able to pinpoint which composers were transition composers, meaning they have strong correlations between their own time period and either the time period before or after theirs. Overall, Beats in Bytes accomplished many of its original goals and expanded beyond what was originally intended with the correlation formula and transition composers.

Approach

In order to extract information from music, we needed a way to represent melodies as text. This had been done for us by the Humdrum Project, created by David Huron, with something called the kern file format. Using these text files enabled us to use the Bash programming language to collect information from music files. However, we first had to find kern files. Though kern is not a common format, MIDI, another music file format, is nearly universal, so we obtained MIDI files from online libraries such as *kunsterfuge* and *Musescore.* Since there was no way to directly convert MIDI to kern, we first converted the MIDI files to musicXML and then to kern. We were analyzing only melodies, so we deleted all but those lines of the music. To extract data from the songs, we wrote a set of programs in the Bash programming language, which we used because it is compatible with kern. Not only were programs written to obtain data, but additional code was developed to organize files and automate various tasks. We then ran these programs on all 500 of our kern files, thus creating a large amount of data. The total output of the Bash scripts included nearly 130,000 data points across 236 metrics.

Results

There were two main parts to the results of our project. The first was analyzing the data we had collected, and the second part was figuring out a way to quantify how close two songs were. These are presented here as two sections to our results.

The term "metric" shows itself throughout the following section. The general definition of this word is "a system or standard of measurement," which is consistent with our usage. However, in the context of this project, it is better defined as an aspect of music. For example, the average length of a note in a song is a metric.

Results - Part 1: Analysis

Data analysis was essential to our project as it was statistical in nature. We analyzed each aspect of our songs individually. In Google Sheets, we found measures of center and variation, conducted t-Tests, and performed other various statistical tasks. Additionally, we found power laws and other distributions using the Casio Keisan online regression calculator. Due to the enormity of both our data and our results, we have selected the most important results to showcase as they clearly and accurately represent our project.

Every song has at least one key signature, which determines the pitches in the song. After finding the occurrences of each key signature, it became evident that Benford's Law fits very closely with our data. Benford's law is a common distribution concerning the occurrences of data points. The equation describing the values of Benford's law is:

$$p = log(n) - log(n-1),$$

where *n* is the usage rank and *p* is the percent of data in that value. Usage rank is the ordinal number that corresponds with the occurrences of a value in a dataset. Using this system, a usage rank of 1 would be most common, and a usage rank of 3 would be third most common.

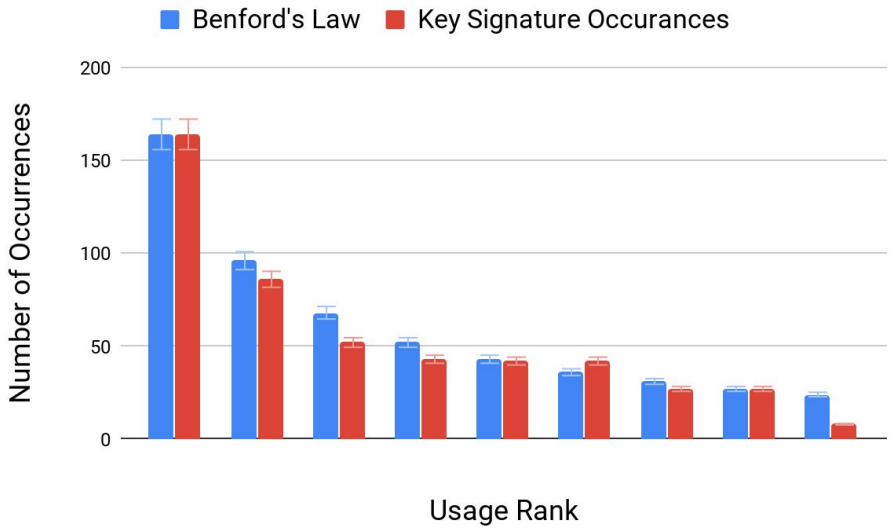

Fig 1): Key signature occurrences vs. Benford's law. This graph shows the relationship between the occurrences of the nth most used Key Signature and the values predicted by Benford's Law. As can be seen, key signature occurrences fit Benford's Law nearly perfectly.

We also collected and analyzed data on note value, which is the duration of a note. One of the many examples of the uniqueness of the Renaissance time period was found in the average note value in the time period. The notes it used were significantly longer than any other time period. This may have been influenced by the slow religious music during this time period.

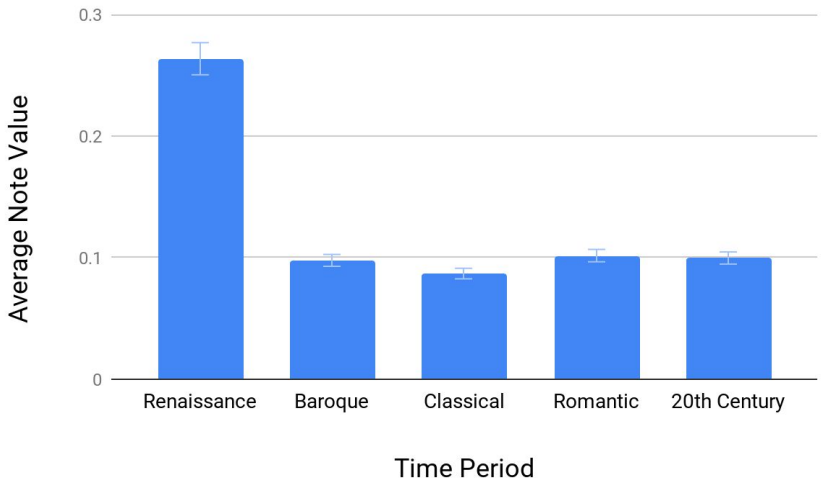

Fig 2): Average note value of time periods. On average, much longer notes were used during the Renaissance than during any other time period.

Another thing we analyzed in music was themes. A theme is a musical pattern repeated at least twice in a song. We looked at many aspects of the themes, one of which was number of times the most common theme was repeated. When we graphed this in relation to the length of the theme, we found a very pronounced stepping pattern. This was most visible within the Romantic time period, as displayed below.

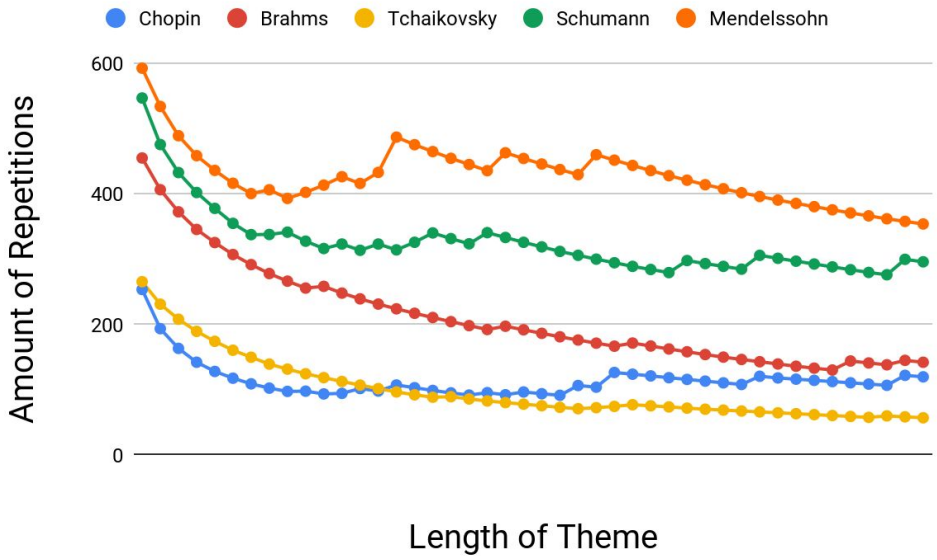

Fig 3): Repetitions vs. length of theme. The graph sees a strange stepping phenomenon. We have selected the Romantic composers for this graph, as it was most pronounced in that time period.

What was especially strange was the near-linear decreases before a sudden increase, as seen most evidently in Mendelssohn's work. The places where the jumps occurred and the magnitude of the increases seem to be inconsistent across multiple composers. Additionally, this stepping phenomenon only occurs with most composers, but not with a select minority. Such composers include Tchaikovsky, who displays little of the phenomenon. Some variation of this stepping pattern appeared in all of the time periods, but the Renaissance time period displayed it the least.

Pitch is how high or low a note is. When analyzing the average pitch of a song, we found that all of the time periods had roughly the same mean. We also looked at their distribution. The data was hardly skewed at all, appearing most often right around the middle.

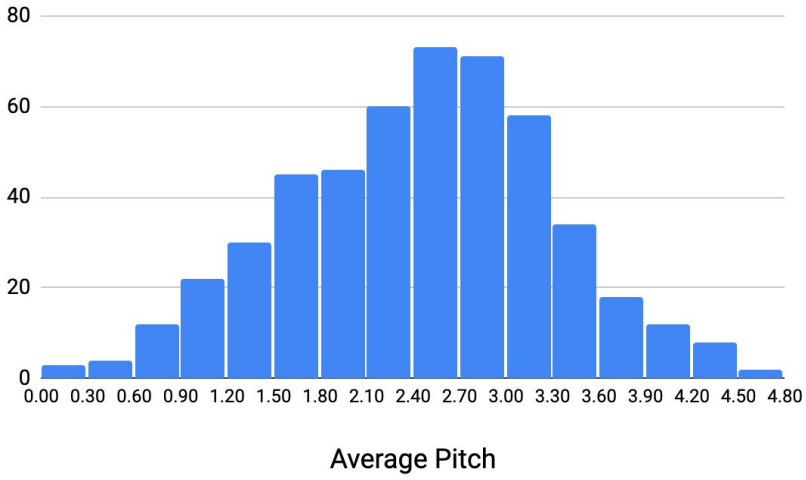

Fig. 4): Distribution of the average pitch. This histogram exhibits a near-perfect bell curve, with small skewness and kurtosis values. The pitches stay below 4.8, which is only the lower half of the possible values.

We additionally studied the pitches that were most used in each song. In this respect, Renaissance almost exactly fit Zipf's Law. Zipf's Law is another common distribution law that is a relationship between usage rank and the amount of data in a specific value. The equation for Zipf's Law is:

$$y = \frac{n}{x^a}$$

Where *y* is the frequency of the value with a usage rank *x,* and *n* is the frequency of the most common value. The value of *a* is equal to 1 in a perfect example of Zipf's Law, but most of the time in the real world Zipf's Law does not have an *a* value of one.

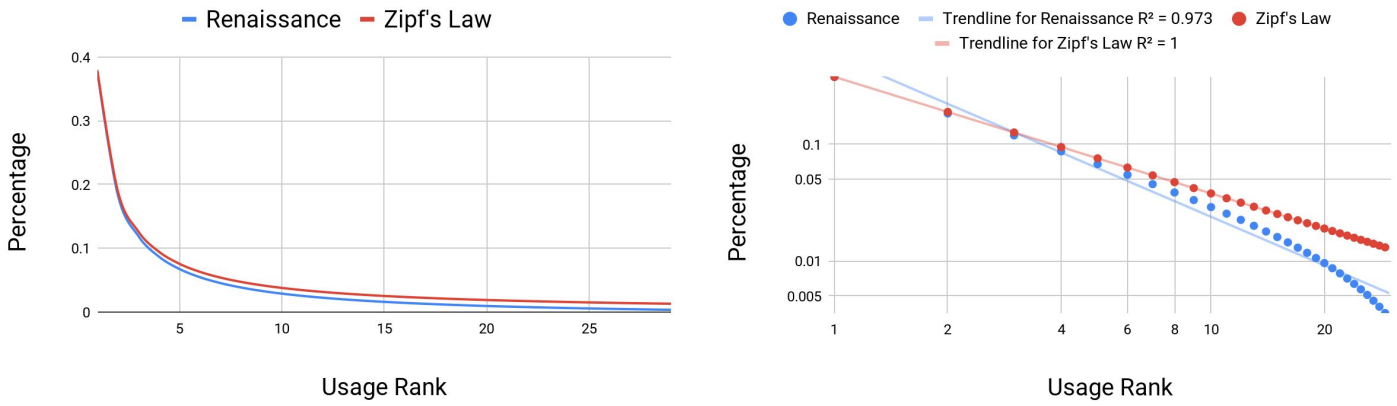

Fig 5): (Left) Renaissance most used pitches compared to Zipf's Law. Blue is the curve of best fit for the most used pitches of the Renaissance time period and red is the values Zipf's Law predicts. The curves were almost identical, with an a *value of 0.992. The curves have very high correlation. (Right) Renaissance vs Zipf's law on a log-log plot. This scale exaggerates the differences of the equations for comparison purposes. The coefficient of determination* (r^2) *is simply a measure of how well a set of data points can be represented by a curve. Here, we can see that the coefficient of determination is very high, at 0.973, showing that our data strongly correlates with Zipf's Law.*

We also analyzed for the 80/20 rule, also called the Pareto Principle, for the most used pitches of each song. For every composer and time period, we took the most used 20% of the pitches (the top 3) and found how many songs used those pitches as their most used pitch. If those 20% of pitches accounted for around 80% of the data, then they followed the 80/20 rule. We found that while only 7 of the selected composers fit the 80/20 rule, all but one of the remaining composers undershot the 80/20 rule (the top 3 pitches accounted for less than 80% of the songs). The exception was the composer Haydn, where his top 3 pitches accounted for 90% of the song's most used pitches. Another composer called Byrd had over 80% of his songs having the same most used pitch, unlike any other composer.

Results - Part 2: Formula

As a large part of our data analysis, we constructed a formula that finds the correlation between two datasets and weights it using the correlation coefficient for each metric.

$$\sum_{i=1}^{n} \left(\frac{|r_i|}{\sum_{k=1}^{n} |r_k|} \right) \left(1 - \frac{|a_i - d_i|}{\sqrt{a_i d_i}} \right)$$

The purpose of the first term is to weight each aspect of music according to how well it fits a trend. The formula starts by taking the correlation coefficient of each metric. It then finds the fraction total that is each coefficient. If the correlation coefficient, *r*, is higher for a metric, that metric gets weighted more.

The second part of the formula depends on what values are being compared. We take distance between them, and divide it by the geometric mean. We chose geometric mean because we were working with ratios. This value is the correlation distance from 1, so we then subtracted it from 1.

For every metric, the two parts are multiplied together, and then summed up to arrive at a final correlation. Due to the weighting process, the output of this formula can never be greater than 1, though it can be negative.

We used the above formula to compare every possible combination of composer and time period, and give values showing the correlation between the pairs. We then graphed the correlation of the music of a time period to the music of each other time period. We could then use this data to graph the usage of the style of a time period over time. The equations for each of the time periods fit the general trend of most of the composers during that era, but they don't fit perfectly. For example, Romantic and Classical composers are best fit by an inverse regression, whereas their curves as time periods are better represented as quadratics.

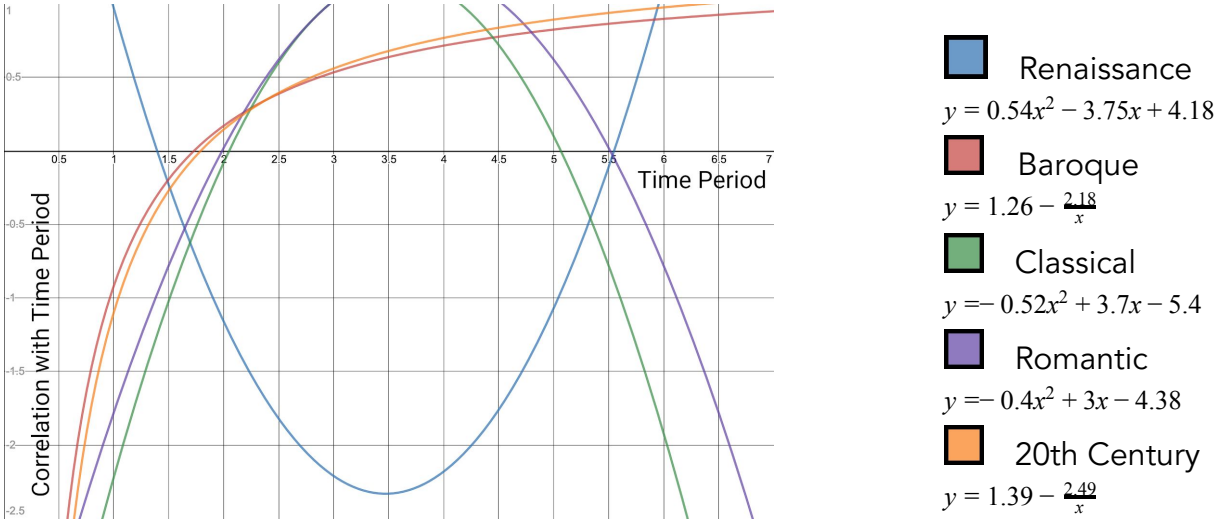

Fig 6): Each curve represents the correlation over time of a specific time period. The x-values of 1-5 on this graph are the five time periods in chronological order, and the corresponding y-values with those on each equation are the correlations with those time periods. Renaissance is best represented by a quadratic with a positive coefficient, meaning that classical music will become more correlated with its style if current trends hold.

We also used this formula to find transition composers, which are composers that bridge the styles of two time periods. Our sample included two: Beethoven and Monteverdi. Beethoven, a Classical composer, had a higher correlation with the following period, Romantic, than his own time period. This supports the commonly held belief that he was a began the style of the Romantic period, and further shows our equation's reliability. Monteverdi was a Renaissance composer who also had a stronger correlation with the following musical period than his own. This was not at all expected and also makes Monteverdi a bridge between the Renaissance and Baroque time periods.

This formula is not, however, exclusive to musical analysis. It is essentially a new type of statistical test. It is similar to a t-Test in the way that it compares the similarity of two datasets, and is similar to correlation coefficients in the way that it outputs a value within a certain range that came from two variables. It weighs different metrics based on how important (and correlated) they are, thus accurately comparing two different datasets, and can be extended to as many metrics as needed, provided the two (or more) sets have the same number of elements.

Conclusions

The most noticeable conclusion we arrived at from our results was that Renaissance music differs greatly from the other time periods. Using the formula that we created, we also found that classical music is becoming more like the Renaissance style. In addition, we discovered that distribution laws that occur in many real-world scenarios are also evident in music. Our results showed that new statistical phenomena also appear in music such as parallel curves, stepping patterns, and power laws.

The equation that we developed not only has applications within the analysis of classical music but can also serve as a new type of statistical test that is powerful, broad, and applicable in a variety of situations. It is a unusual test due to its use of weighting to make outputs more accurate and is the first of its kind to harness the power of trends in calculating correlations.

Aside from the statistical discoveries made thus far, there is even more potential for this project. To start with, we would like to analyze many more songs than the 500 we have as of now, perhaps analyzing all the songs in the various online music file libraries we used. We could split classical music into far more periods than we did here, and add more composers as well. The data could even be organized geographically, in order to find trends based on country and culture. We are also looking to add more metrics and cover some of the ones we missed in this project, and not only polish our scripts but modify them as well to accept multiple lines of music (and not just the melody). We could also expand on these libraries by adding the several hundred kern files we have made, making it easier for future researchers who wish to use the same database. Additionally, we would like to bring our scripts and analysis results to the wider world. We already have a Github repository, and are in the process of creating a website. This website would showcase our results, while embedding an application that would allow people to analyze music files in an intuitive, simple way. Our research occupies a key gap in the field of musical analysis, not only analyzing hundreds of properties of classical music for various statistical trends but also simplifying the entirety of classical music with a direct formula. Our approach to the analysis of classical music has brought to attention a new method, one of applying code to what is thought of as qualitative arts, breaking it down and analyzing what is usually accepted as a very imprecise and human area of culture.

References

(2017, Feb). *Bash Script - Variable content as a command to run.* Retrieved from https://stackoverflow.com/questions/5998066/bash-script-variable-content-as-a-command-to-run

(n. d.). *CCARH Humdrum Portal.* Retrieved from http://humdrum.ccarh.org/

(2019, 8 Mar). *Circle of Fifths*. Retrieved from https://en.wikipedia.org/wiki/Circle_of_fifths

(2019, Feb). *Count multi-line patterns in a file.* Retrieved from https://unix.stackexchange.com/questions/496773/count-multi-line-patterns-in-file

(2019, 4 Mar). *Degree (music)*. Retrieved from https://en.wikipedia.org/wiki/Degree_(music)

(2019, Feb). *find second largest value in array*. Retrieved from https://unix.stackexchange.com/questions/496213/find-second-largest-value-in-array

(2019, 21 Feb). *Pareto Principle.* Retrieved from https://en.wikipedia.org/wiki/Pareto_principle

(n. d.). *The Humdrum Toolkit: Software for Music Research.* Retrieved from http://www.humdrum.org/

(2019, 11 Jan). *Zipf's Law*. Retrieved from https://en.wikipedia.org/wiki/Zipf's_law

Bellos, A. (2014) *The Grapes of Math - How Life Reflects Numbers and Numbers Reflect Life*.

Bourne, M. (2018, Apr. 11) *Graphs on Logarithmic Scale and Semi-Logarithmic Axes.* Retrieved from https://www.intmath.com/exponential-logarithmic-functions/7-graphs-log-semilog.php

(2012) *Music Theory Fundamentals - Key Signature.* Retrieved from http://musictheoryfundamentals.com/MusicTheory/keySignatures.php

(2019, Feb. 26) *Benford's Law*. Retrieved from https://en.wikipedia.org/wiki/Benford%27s_law

Khan, S. (2017, July 11) *Calculating Correlation Coefficient r.* Retrieved from https://www.khanacademy.org/math/ap-statistics/bivariate-data-ap/correlation-coefficient-r/v/calculating-correlation-coefficient-r

Khan, S. (2016) *Calculating Standard Deviation step-by-step*. Retrieved from https://www.khanacademy.org/math/probability/data-distributions-a1/summarizing-spread-distributions/a/calculating-standard-deviation-step-by-step

(2003) *shell_exec.* Retrieved from http://php.net/manual/en/function.shell-exec.php