Big Data, Big Issue? : An Analysis of Different Software to Tackle Big Data

Rewa Sood, Yuhe Zhu, Rong Rong
**Introduction:**

Big data is a relatively new concept in many fields and requires new methods to tackle the challenges it brings. The facets of big data that pose as the greatest challenges are the volume, variety, and complexity of data. The internet itself provides an access to an unprecedented amount of data, and other tools for recording and measurement only add to the sea of numbers and letters. The key is to wade through this volume and collect only the significant pieces of data.

The word significant is highly subjective and each individual tackling big data has to formulate their own definition. However, creating this definition requires the understanding of the underlying and interconnected currents in the data. The first step is to compile the various formats of data into one manageable form. Having the data grouped together based on something even as basic as a label or a count allows for a more clear view of the data, and even the underlying patterns.

At this point, the data needs to be processed in order to find various patterns and test their significance. The challenge here is to find the most effective software that will provide the functionality required to meet the specific needs while also not being unnecessarily complicated. The goal of our analysis is not to find the “best” software for big data, since each type of software provides different tools, which fit better or worse depending on the need. Rather, our goal is to display the benefits and setbacks of each software in an unbiased way. In the end, the goal of manipulating big data is to convert quantity to quality, and our goal is to find in which cases each software leads to the path of least resistance.

We conducted our analysis through the Rossmann store sales data provided by Kaggle.com. The data was provided in two sets. One set gave information about the dollar amount of sales, the number of customers, whether the store was open, whether the store had a promotion, and whether the area had a school holiday, a state holiday, or both. This information was provided for over a thousand stores for each day of the week for many months, resulting in over a million data points. The second set provided information about the type of store, the distance and number of the nearest competitors, and year since the competitor store opened. The second set provided over a thousand data points, which were entirely different from the first set.

We analyzed these two data sets in Microsoft Excel 2010, Python, R, and SAS. Each method of analysis had its own drawbacks, however the main issue throughout was handling the more than million data points and dealing with missing data. We will first discuss each software’s pros and cons and end with our approach for dealing with missing data and further steps we would like to take based on our work so far.

**Excel:**

Microsoft Excel 2010 allows data to be formatted as a spreadsheet for straightforward visual of the data along with data manipulation tools without the requirement of an extensive
programming background. However, the program makes it exceedingly difficult to load large files, such as the million plus Rossmann data points. Additionally, Excel’s data manipulation tools, such as the pivot table, are incompatible with large amounts of data.

Table 1 below shows an example of the layout of the Excel spreadsheet when the data is loaded into the program. The stores are numbered from 1 to 1115, the days of the week are numbered 1 through 7 in order, sales are given in dollars, and the last four columns are indicated active by a 1.

**Table 1: Excerpt from Excel Spreadsheet**

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5</td>
<td>5263</td>
<td>555</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>5020</td>
<td>546</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>250</td>
<td>2</td>
<td>9536</td>
<td>749</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>500</td>
<td>6</td>
<td>5592</td>
<td>400</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>500</td>
<td>7</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1000</td>
<td>3</td>
<td>5535</td>
<td>544</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1115</td>
<td>1</td>
<td>3697</td>
<td>305</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Displaying all of this information for over a million data points-1017209 to be exact- is unpractical. All of the Excel work was completed on a 8-GB RAM, Intel i7 core, 64-bit operating system, comparable to most laptops currently available. These specifications did not provide enough power, as Excel repeatedly crashed while trying to load the data. A more advanced processor would be required to analyze this data more deeply using Excel.

One way to circumvent the roadblocks above is to use tools that allow analysis of certain sections of the data at a time. These tools do require that the computer is able to load all of the data into excel first. Table 2 below shows an example of a Pivot Table. This table shows the total sale amount in dollars for certain stores.

**Table 2: Excerpt from Pivot Table**

<table>
<thead>
<tr>
<th>Store</th>
<th>1</th>
<th>250</th>
<th>500</th>
<th>750</th>
<th>1000</th>
<th>1115</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total</td>
<td>3716854</td>
<td>6615557</td>
<td>3330211</td>
<td>3108353</td>
<td>4635525</td>
<td>4922229</td>
</tr>
</tbody>
</table>
Figure 1 below shows the entire layout of a Pivot Table, the excerpt of which is in Table 2. The Pivot table fields allow various aspects of data to be compared against each other. Placing one aspect in the COLUMNS field and the other in ROWS compares the two to each other. In addition, fields can be placed in SUM VALUES so that certain aspects can be compared to the sum of others. Figure 1 shows how each store can be matched to the sum of its own sales.

Figure 1: Pivot Table

While pivot tables allow for rapid comparisons of various facets of the data, certain aspects cannot be compared in a pivot table due to the memory constraints of the device. Generally, the table is only useful for comparing one aspect with the sum of another, such as Store and Sales or Days of Week and Customers.

In all, while Excel requires the least experience with big data and data manipulation to be able to start exploring the data, the sheer volume of data limits its usefulness\(^3\).

*Python:*

Python is a programming language that allows data manipulation through the execution of various scripts. Python can handle a lot of data, however using Python effectively requires learning how to code in Python and a lot of time spent debugging. Unfortunately, a lot of our work with Python was put into overcoming many issues we had with the software. Therefore, all of the pros for this software are derived from tutorials and other websites which denote how it should work. The cons come entirely from our experience.

Python allows programmers to manipulate a large amount of data via its many packages, specifically Numpy, Scipy, matplotlib, csv, Pandas, and scikit-learn\(^4\). Data in Python is generally
treated as an array of values, whether they are integers or characters, and each array must be of
the same type. Numpy is used to manipulate these arrays of data. Treating data as an array of
values makes ordering the data based on any criteria easy because each value is indexed. This
also eases data cleaning as each missing data point is also indexed.

Scipy contains additional mathematical functionality such as linear algebra for array
manipulation that complements Numpy. matplotlib is Python’s plotting package. For our
application its main use would be to plot histograms and scatterplots of different combinations of
data. Both histograms and scatterplots can be used for quick identification of trends in the data.
The csv package is used to import database and spreadsheet files into a Python interpreter.

Pandas gives programmers the ability to use Python as a data analysis tool rather than just
a data manipulation tool. The use of Python with the Pandas package is comparable to using R.
The benefit is that once the data has been imported and cleaned, the analysis can be done in
Python instead of having to import the manipulated data to a different language. The
scikit-learn package is used entirely for its RandomForestClassifier class. A random forest is a
collection of decision trees created from a random assortment of different facets of the data. Each
tree outputs a prediction value, in our case sales in dollars, given a certain input, such as day of
the week, or competitor’s location. The data can be put through the trees until the tree with the
best prediction is found. One drawback is that the tree with the best prediction may not give the
most accurate results, since the tree may overfit the sample data.

Python is an excellent tool for handling big data since, as seen above, its many packages
are tailored to manipulating data without seeing all of it or even pairing subsets of the data for
comparison manually. Numpy, Scipy, csv, and matplotlib provide all the tools required for
importing and cleaning the data, without crashing the program or requiring an excessively
powerful computer. Pandas and Random Forests allows for data analysis without needing to
switch over to other programs such as R.

On the other hand, in order for Python to be an effective tool, the individual using it has
to learn how to program in the language first. The Kaggle website, referenced previously,
provides a lot of code that beginners can use and learn from. The beginner is then limited to the
functionality given to them. However, there are a myriad of resources on the internet to learn
enough Python so that the programmer can create their own functionality. Taking the time to
learn the language takes time away from playing with the data and testing various hypotheses
regarding underlying patterns. Given a limited amount of time, the programmer is forced to
decide whether they need to learn more Python or use the skills they already have to analyze the
data, if they can.

Another issue is the various flavors of Python interpreters a programmer can use. Kaggle
recommends Anaconda while other popular versions are Canopy and Jupyter. Each interpreter
has its own layout and transferring code between interpreters can lead to errors due to differences
in formatting. This problem requires that everyone working on a particular project use the same
flavor of Python interpreter.
The major problem we ran into however, was importing the csv files of data into the interpreter. The issues were getting the interpreter to find the file since the interpreter required a very specific file address. Additionally, the Rossmann data was a mixture of integers and characters which does not fit the requirement that an array have a single type. Thus, when the file was found, the csv reader could not convert the file to a usable array. In order to overcome this obstacle, the characters had to be converted to integers in some ordered way or removed entirely which may negatively affect the results. Overall, Python is a fantastic data manipulation and modeling tool for those who know how to use it.

R:

R is a widely used software for statistics and data science. For our analysis of the Rossmann datasets, we used RStudio, which is an integrated development environment for R. During our first-stage analysis, we used R to visualize the “train.csv” file, which provided data that we could use to create a model, and experienced a few of its advantages with regard to big data. R is a free software, which allows convenient downloading and installation. First-time users can access the software easily and may try R without a budget burden. Since R is open source, users and developers can upload packages that are designed for various data science techniques. In this way, R becomes powerful in statistical analysis and possesses up-to-date packages for new statistical trends, such as random forests and neural networks. Although there are plenty of packages built in, the syntaxes between different packages are generally uniform. Before utilizing a new package, the user only needs to install the corresponding package and then use it as usual. If the user is not sure about which package to choose, websites such as R-Blogger and R-Quick can be really helpful to help tackle different datasets. These websites are clearly stated and full of examples for users to learn and extend. Implementing R on different systems, such as Linux, Mac, or Windows also does not cause syntax confusion. The language in R is a relatively high level language and tightly related to statistical abbreviations. If familiar with statistical terms, the user should be able to quickly identify and handle the syntax in R. When importing “train.csv” into R, it took a few seconds and the imported dataset was already organized in columns. R generally ignores missing values. For our dataset, this feature provides convenience while also decreasing the accuracy of the data. Figure 2 below is an example of the RStudio interface and the table of “store.csv”.

R also has disadvantages. It may be very slow when dealing with large amounts of data. The “train.csv” data set took a while to process, when plotting certain variables against the sales. Another confusion about R is the package installation. Certain functions are built-in under certain packages. If the user does not have the proper package installed, errors occur without any clues to install the needed package, even if the syntax of the function inputted is obviously correct. R is not expert in managing, creating, or merging datasets. Datasets imported into R should already be roughly clean and ready for statistical analysis. However, when manipulating
with big data, datasets may not be constant all the time. Updating datasets thus become an issue in R.

Figure 2:

**SAS:**

SAS is another software for analytics, which provide different versions to accommodate the needs of “individuals, small and midsize business, enterprise, academic, government”. 13 To prepare the Rossmann dataset for later process in Python, we used SAS to merge the datasets “store.csv” and “train.csv” by Store ID; we chose SAS to do the merge because SAS is suitable software for dealing with data reorganization.

SAS basically involves two types of procedures, the DATA procedure and the PROC procedure. As the name suggests, the DATA procedure is used for creating and changing datasets, and the PROC procedure is good for calling certain datasets to process. Compared with Python, the language in SAS is pretty simple and easy to understand. Most of the languages are abbreviations in human language and scientific terms. For example, the PROC procedure used for printing begins with “proc print”. SAS has an advanced error detection system. If there is an error, SAS gives specific diagnostics and suggestions about where to modify. Since SAS company manages SAS, the company is able to provide professional post-service and assistance. SAS has built in buttons to input csv files into SAS and change it into a SAS file, which saved us
time. The PROC procedures have many functions to perform statistical analysis and provide neat outcome.

We used SAS to merge the datasets from the Kaggle contest, but SAS is not always expert in merging. In our case, the merge is a typical one-to-many merge, because the Store IDs in the “store.csv” are different for each entry, whereas the “train.csv” contains several observations for the same store. After the datasets are sorted by the merging variable, SAS has the ability to merge the datasets together by automatically deciding where to start or end. However, if the user needs to merge datasets that both have multiple observations for each merging variable, SAS generates incorrect dataset and causes confusion. To overcome the merging problem, the user should use the extension, PROC SQL, in SAS, which is a combination of SAS and SQL. SAS may face obstacles in analyzing big data, due to its low speed. In our example, merging the datasets cost over 3 minutes on a personal computer. This speed problem may be solved when companies use better computers, but the speed is still an issue SAS should improve. Another limitation about SAS is the price. Although academic users may receive free trials, the price for individual users is still too high. The price for organizations can be negotiated, but it is still pricy compared with other software, such as R and Python.

**Data Cleaning:**

It is necessary to think why data may be missing while dealing with missing data. In Statistics, there are generally two scenarios, “missing at random” and “not missing at random”. The missing data that are said to be “missing at random” are those unrelated to actual values. In other words, these data may not be important. Even though the data size would be smaller than the original data set, ignoring the missing data will not lead to a biased result. On the other hand, if the missing data are related to the actual values, they are called “not missing at random”. In such situation, analysis that is based on available data alone will typically be biased.

In our project, we used two ways to deal with missing data. One is for the data that exist in the testing data but not in the training data, where we created a new indicator **Missing** in test data to show whether this situation happens. The other way is for the data that are missing in both test and train data. We have tried two methods for this scenario, replacing missing data with the average and simply delete the whole entry. Both of these two methods were based on the assumption that the missing values in our project are “missing at random” and will not be important to affect the actual values.

**Variable Selection and Combinations:**

In this project, we learned the importance of feature selection. We first replaced the “Store ID” with “Store Type” as the response in our model so that we could predict the sales based upon the type of a store, a larger variable from which we could collect more information. Then, we changed the format of the predictor “Date” from Character to the Standard Date
Format in R by using functions to extract “Year”, “Month”, and “Day”, three variables from “Date”. From this step, we eventually generated a critical variable “Week”. Since sales in each store were closely related to holidays, it made more sense, intuitively, to predict the weekly sales instead of sales per day.

In terms of the variable combination, there were many predictors in the original data, including store number, days of the week, indicators about promotion, holidays, and whether the store was open. We initially used all the predictors in our model, but the VIFs (Variance Inflation Factor) suggested that there was a multicollinearity issue between “Open” and “StateHoliday”. Based on this information, we refitted a smaller model with “Sales” as the response and only “Store”, “Week”, “Promo”, and “SchoolHoliday” as predictors. The Multiple R-squared dropped while the Adjusted R-squared increased. From this, we learned that despite the reduced margin of error when a model involves more predictors, adding peripheral predictors will not do much to improve the model in that the extra predictors will increase the variation of the estimates and make the model harder to interpret. The F-test also showed that we preferred the simple model at the 5% significance level.

**Conclusion and Next Steps:**

Initially, the goal of our project was to find the best software for big data. After working with the various software, we realized that there was no one best software, rather each software had its own strengths and weaknesses, and each was more or less useful depending on which aspect of data the researcher wanted to focus on. Excel is best for the individual who does not have a strong programming background and who does not have an overtly large data set, on the scale of a million data points. Python requires programming knowledge, while also giving the flexibility to manipulate large amounts of data quickly and efficiently. R and SAS both depend upon a working statistical knowledge and are slow when processing large amounts of data, however they also provide many tools to perform tests determining the significance of the variable combinations chosen in various trials. In all, we found that the researcher would be best served in first determining what her needs are and then deciding which software would best meet those needs given its particular strengths and weaknesses.

The focus of this project was on evaluating various software’s ability to handle large amounts of data. The next steps of this project would be to analyze a set of data using each software while keeping our findings of each one’s strengths in mind. This would allow us to determine not only how each software handles the data, but also how easily and effectively an individual can create and apply models on that data set.
Works Cited:

5. https://pypi.python.org/pypi/numpy
8. https://docs.python.org/2/library/csv.html
14. http://handbook.cochrane.org/chapter_16/16_1_2_general_principles_for_dealing_with_missing_data.htm