STA 160 - Practice in Data Science

Rani Bindra
Jingyi Wang
Mitchell Layton
Luna Qiu

---

# Using Optical Character Recognition to Extract Wine Names and Prices From Images of Catalogues

GitHub URL: https://github.com/MitchellRLayton/STA160-Wine-Project

URL for project Web site: https://sta160wineproject.weebly.com

## Abstract: Executive Summary

For this project, we were given approximately 4,000 scanned wine and alcohol catalogs from the 20th century, which were printed on paper and likely never existed as a digital text before printing. Therefore, our goal for the quarter was to digitize the scanned images somehow, and extract the alcohol names and prices (per bottle and per case). Our method of completing this task was to test different optical character recognition (OCR) softwares for the most accurate textual output.

The two different OCR methods we chose to use were PDFPenPro (PPO) and Tesseract. There were various conditions where each software gave better results, therefore we decided to use both methods and compare the output accuracy. We then wanted to determine if we could rule out one of the OCR softwares altogether, or create a set of rules as when to use one over the other. This was accomplished by collecting and formatting all of the wine names and prices from both softwares and comparing it to our own created training sets from the original catalog images, which we will refer to as truth sets.

Our main goal using the two OCR methods was finding the method that outperformed the other and figuring what the circumstances were for each case. For some images, the desired text was found in table and column like formats. Conversely, there were some images with very unpredictable formats, where the names and prices were not easily extractable by a general approach solution. For instance, irrelevant textual elements and randomly located columns were very difficult for OCR to work promptly and efficiently, so we decided to focus on the more neatly formatted and tabular images. Therefore, we created an algorithm to extract information from the Rtesseract results which largely depended on knowing the coordinates of the tables on an image. The PDFPenPro general approach required implementing code that would grab all of the textual results and extract the names and prices of alcohols via careful string processing/parsing.

We then used our truths set to compare the output we received on the images through the two OCR methods, which we will go into detail later in the report. If we only compare how well the two OCR methods does on the text, we found that overall, PDFPenPro was accurate 91% of the time and R Tesseract was accurate 65.0% of the time. However, unlike with R Tesseract, the order of text from PDFPenPro was not the same as the original image.

## Introduction and Motivation

We were interested in extracting alcohol names and prices from the scanned catalogs so that this information could be used in the future for marketing or other economical use. We also wanted to develop a method that was more efficient than having people parse through each catalog by hand, which also increasing the likelihood of human error that would be made when copying the information. Extracting text from scanned images using OCR software has advantages. Not only does it save on time by not having to develop our own text extraction software, but it also provides its users with good functionality and flexibility.

Our project revolved around using two methods: utilizing RTesseract to strategically extract text, and using the results from PDFPenPro which we were given from our professor. The motivation behind using the results from both OCR methods is that we would be able to directly compare the both outputs. Through both extraction methods, we would be sure to make note any instances where the softwares were doing really well, and where they did poorly so that we could identify the trends of both softwares. Finding the situations and conditions where the softwares performed well was important because this would allow us to make the decision of using one software over the other in certain situations.

## Project Design

The project tasks were outlined so that three out of the the four total group members would work with RTesseract in order to extract all of the relevant data. The fourth group member would work with the PDFPenPro results. Since we had access to a sample of 85 PDFPenPro image results, which were deconstructed paragraphs or tables from the images and put inside of a Word document, it made sense to only put one person in charge of extracting all of the relevant information via string and text processing. Therefore, the other three group members worked with RTesseract, since we had to develop our own strategic methods of text extraction, and implement any text processing procedures from there.

## Implementation & Data Summary

### Data

The data used for this project was a collection of 4,111 .jpg images of scanned alcohol catalog. We worked primarily with a random sample of 100 images out of all the ones we were given. Because these images were scanned, many of them were prone to slight rotations, or had the margin of other catalogs other page included in the scan, which complicated the OCR techniques we were using. While a majority of the catalogs included the information that we

wanted to extract, there was a subset of them that were irrelevant to our project. Such images included only pictures or graphical designs without any textual information.

The PDFPenPro software found the textual and/or table elements of an image and returned results which were extracted to a Microsoft Word document. The Rtesseract results from our algorithm returned the wine names and prices from the tables in each image we looked at. We will continue outlining the process of using PDFPenPro and Rtesseract below.

**PDFPenPro method and procedures**

We started by extracting all of the paragraphs and tables from each document. In order to get the most accurate text output for our final accuracy comparison, some formatting fixes had to be implemented on certain result strings. This included developing functions that iterated on the text in order to remove any special characters. These functions also fixed incorrect price formats since there were times where the software would omit a decimal point or put them in the wrong spots, giving the incorrect price. PDFPenPro would also fix instances where catalogs had the names of the alcohols followed by a long line of dots that would lead up to the prices by removing these consecutive periods. Many little formatting issues such as these were taken care of before the name and price extraction, which was an advantage of this software.

For organizational purposes, it was important to add the labels of the Word documents with their corresponding result sets, by creating a python dictionary with each key being a Word file name and each item is its corresponding results. The result sets consisted of two labels: text results, and table results. If there were multiple paragraphs and/or tables within one document, it would aggregate all of the text associated with either the paragraphs and/or tables and would add it to a list associated with the text result and/or table result respectively. When there were no paragraphs, tables, or neither found, the software would set the result to “None.” After establishing this data structure of the  PDFPenPro results, we would iterate over the results again using developed functions in order to process each result set to its final desired output (alcohol names and prices). This processes can be show below:


```
{'UCD_Lehmann_4043.docx': [{'TEXT': ['JURA WINES 874 1012 RIESLING SUPERIEUR 1966 Nicolas 1.89 20.40 728 TRAMINER 1967 St Od
ile 2.39 25.80 704 ALSATIAN PINOT NOIR ROSE 1967 St Odile 2.49 26.90 718 GEWURZTRAMINER 1967 St Odile 2.49 26.90 COUNTRY WINE
S Here are the very wines that you encounter when you tour through the countryside of France 5000 REGENT COTES DE BERGERAC Mo
elleux Nicolas 1.29 1.29 13.95 997 COTEAUX DU LANGUEDOC ROUGE Herault 13.95 904 COTES DE PROVENCE ROUGE 1967 Nicolas Aix-en-P
rovence 1.49 16.10 856 SCIATINO BLANC 1966 Corsica 1.49 16.10 847 SCIATINO ROUGE 1966 Corsica 1.49 16.10 520 CORBIERES CHAINT
RUEL 1966 Nicolas 1.49 16.10 800 ST POUCAIN-SUR-SIOULE Bourbonnais Dry White 1.59 17.15 810 FITOU GRAND VIN ROUGE DU ROUSSILL
ON Pyrenees .69 18.25 805 CHATEAU ROUBAUD ROUGE COSTIERES .79 DU GARD 1966 885 COTES DE VENTOUX ROUGE 1967 Gold Medal .79 Pro
vence 19.35 808 CAHORS ROUGE 1966 Lot Valley 765 APREMONT 1967 Domaine Boniface Savoie 21:1 2.19 688 CASSIS BLANC DE BLANCS 1
967 2.29 23.65 Domaine du Paternel 581 CREPY 1967 Asher-Storey -.15 2.49 807 BANDOL ROUGE 1964 Chateau Pradeaux 2.79 Provence
 -70 782 BOUZY ROUGE 1966 Domaine Vesselde 3.49 Still Champagne 4.99']},
   {'TABLE': ['No 771 746 948 947 CHATEAU DARLAY WHITE 1964 Comte de Vogue CHATEAU DARLAY RED 1964 Comte de Vogue ARBOIS JAUN
E 1955 CHATEAU CHALON 1959 Bottle 3.49 3.49 4.95 7.75 Case 37.70 37.70 53.45 83.70']}]},
```


```
{'UCD_Lehmann_1975.docx': None}
```

The process mentioned above was very important for identifying how well each document was doing in terms of returning accurate or inaccurate results. It also made it easier to find instances where PDFPenPro was returning nothing, even though when we went to check on that image directly, it would be a complete catalog with many names and prices on it. This meant that PDFPenPro was evidently failing on reading certain images, possibly because of not preprocessing the images. Preprocessing the images entailed either changing the orientation of the image, or switching the image from color to black and white to ease the OCR software.

In order to extract the desired output, we needed parameters to index our text extraction methods on. Some methods were easier than others because of the ability to index an entire results string from its bottle number to its prices. Since the desired information would be between the bottle ID number and the prices, we were able to extract all of the necessary information between these two indices. Such example can be shown using the following example:

```
804 CHATEAU CANON St Emilion 5.49 59.30
```

Another instance of extracting the desired format was the case where alcohols did not have a bottle ID number, and their names were in all capital letters. Since early stages of the text extraction process would firstly find instances of prices in a string, it would then become straightforward to extract all of the text backwards, up until the word at the current index was indeed uppercase (JUSTERINI), and the previous word before that uppercased word was not uppercase and/or was a year. Below is an example:

```
JUSTERINI BROOKS 91 proof 4.59 52.33
```

In the development of the text extraction code, it became too cumbersome to use the index of the alcohols year, since there was variation between where the year was located in a string. Occasionally it was placed as a leading descriptor of the name, while other times it was the last item mentioned before the prices.

**RTesseract method and procedures**

The second OCR method we used was Tesseract in R Studio. We used the Tesseract software through the package RTesseract. RTesseract is a package created by Duncan Temple Lang that can be found on GitHub. With RTesseract, the image is read by creating boxes around each word, and then reads the word in the box using Tesseract. The output from RTesseract consists of the coordinates of the boxes, the text in the box, and the probability that Tesseract read the word correctly. We used RTesseract because we needed the additional information RTesseract provides to help us extract the text and then put them into a table. We utilized the coordinates of the boxes to find each line in a table, so that we could extract the wine names and prices. A key insight we got to the data was that when plotting the bottom coordinates of the boxes, each line of text were grouped together, such as in the plot seen in the next page. So, we could separate each line of the table by looking at the difference of the bottom coordinates

between the words next to each other. If the difference was large, then the next word is the start of a new line.

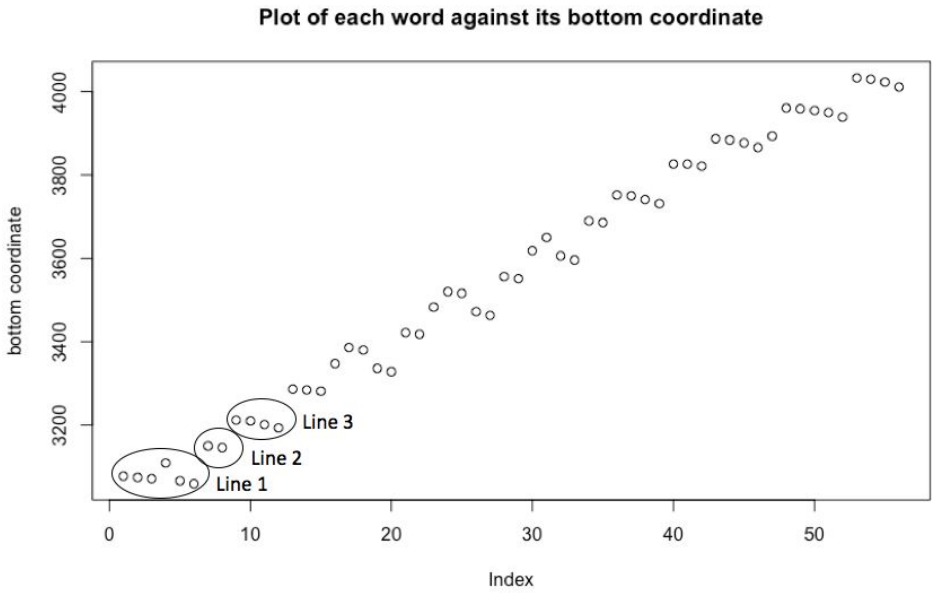



Figure 1: Plot of each word read by R Tesseract against its bottom coordinate

To extract the wine names and prices from the R Tesseract output, we first tried to work solely with the text output, similar to what we did with the PDFPenPro output. However, once we found that we can extract each part of the table line by line, we continued working in that direction. Even then, in most cases, RTesseract gave us very messy results to work with, and sometimes, the results did not closely resemble the text in the images at all. Furthermore, we tried to create an algorithm that will find each table in an image, but did not have enough time to finish writing our algorithm, so we gave up, in hopes to pursue it in the future. When we tested our algorithm to extract the text, we first found the coordinates of the tables for each image ourselves by looking at the coordinates of the text located at the corners of the table. We needed to do this to make our algorithm work the way we wanted it to.

There were four issues we had with RTesseract. The first one was that Tesseract is unable to identify text that is not black. Therefore, we tried to make our images black and white and then read the image through RTesseract. After trying this, we decided not to pursue this because the output was worse than the original image since the software read bits of the image that was not text. The second problem was that most of the images were rotated. Even if the image was rotated only slightly, this reduced the accuracy of the OCR reading. When the images are slanted, RTesseract would split a word into two words. This happened because when RTesseract created the boxes around each word to read, it could only create horizontal rectangles (boxes) around the text. The third problem was that the tables in the alcohol catalogs did not all share the same layout, so the algorithm needed to be changed for each table to be able to process the R Tesseract output.

The fourth problem was that the images also included descriptions of each wine located right below each wine name. Because our algorithm worked by extracting text line by line, and did not differentiate between lines that included only the wine names and prices, and lines that had descriptions of the wines. Therefore we can conclude, our algorithm worked best when the tables in the images were laid out in a structured way - names, bottle price, case price - with no excess information, such as descriptions of the alcohols. We plotted the heights of the boxes (calculated by bottom coordinate - top coordinate) because in the images the wine names and prices tended to have a larger font than the description. However, there was no noticeable difference between the heights of the boxes for words in the description and the words in the names and prices. Because of this and the time constraint of the project, we decided to focus solely on the tables with just the names and the prices.

We tried to produce an algorithm that regardless of the type of table and how much unnecessary information was in the table (such as descriptions of the wines), we would always be able to end up with a dataframe with just the wine names and prices. However, we were not able to do this, so, we had a different code for each image, but still based them off of the algorithm that extracts text line by line.

Because of the issues stated above, in the random sample of 100 images we have, there were only 14 images (14% of images) that had well-laid out tables in which we could extract text from.

After getting the wine names and prices and putting the information into a table format, we needed to compare our output with the truth set to find the accuracy of our output. Therefore, we develop a truth set by hand. The truth set consisted of the text of the tables in the image typed up and in the same structure as our output we got from RTesseract. We calculated the accuracy using the adist function in R, which counted the number of characters that needed to be added, deleted, or substituted from the output to get the truth.

## Results

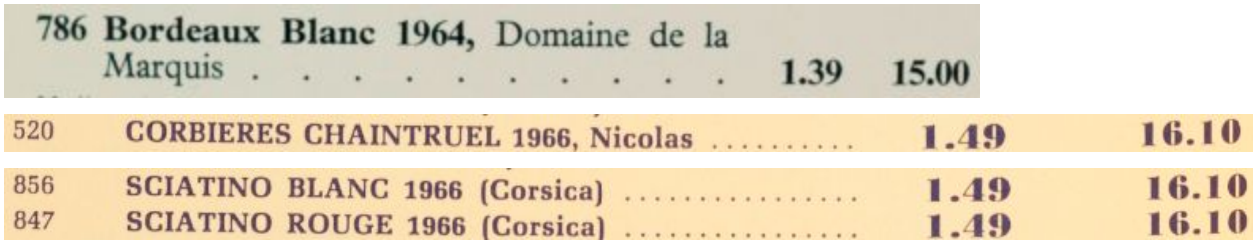
### PDFPenPro Results

Below is an example of the output we got using the PDFPenPro software. After looking at the all of the processed text results from the Word documents, there were enough instances of formats such as this one that warranted implementation that was described above in the methods and procedures. Furthermore, there were many empty result sets for images that had plenty of structured table data in the image, meaning that PDFPenPro may have had trouble with it’s image processing and/or optical character recognition implementation.

| | BottleNo | Name | BottlePrice | CasePrice |
|---|---|---|---|---|
| 0 | 786 | Bordeaux Blanc 1964 Domaine de la Marquis | 1.39 | 15.00 |
| 1 | 520 | CORBIERES CHAINTRUEL 1966 Nicolas | 1.49 | 16.10 |
| 2 | 847 | SCIATINO ROUGE 1966 (Corsica) | 1.49 | 16.10 |
| 3 | 856 | SCIATINO BLANC 1966 (Corsica) | 1.49 | 16.10 |
| 4 | 904 | COTES DE PROVENCE ROUGE 1967 Nicolas (Aix en P... | 1.49 | 16.10 |
| 5 | 511 | LMBFRAUMILCH Kendermann | 1.49 | 16.10 |
| 6 | 800 | ST POUCAIN SUR SIOULE Bourbonnais (Dry White) | 1.59 | 17.15 |
| 7 | 301 | PINOT CHARDONNAY LAMARTINE (Caves Cooperative) | 1.59 | 17.15 |
| 8 | 230 | CHATEAU LAURETAN BLANC 1964 | 1.59 | 17.95 |
| 9 | 793 | BERNCASTELER SCHWANEN 1960 (Dunweg) | 1.69 | 18.75 |
| 10 | 542 | NIERSTEINER DOMTHAL Kendermann | 1.69 | 18.25 |

Figure 2: Code results from a sample of 10 out of the 183 (correct) distinct results returned in this specific format (*bottle no. // name // (year?) // prices*) ordered by bottle price.

Here is a brief snippet of the first 4 results as they would have appeared on the catalog image.




Below are the accuracy results using PDFPenPro output against the developed training sets.

| image | % wrong | % correct |
|---|---|---|
| 0011 | 18.4% | 81.6% |
| 0237 | 5.6% | 94.4% |
| 0551 | 5.0% | 95.0% |
| 0558 | 10.7% | 89.3% |
| 0644 | 8.5% | 91.5% |
| 0869 | 11.7% | 88.3% |
| 3479 | 7.1% | 92.9% |
| 3794 | 26.7% | 73.3% |
| 4086 | 6.1% | 93.9% |

The reason the error rate was so high was because PDFPenPro may not have read certain parts of the information, such as the bottle number (if there was one), or the prices were read incorrectly. For example, there are occasions where the software will misassociate the prices and names of neighboring lines. Other than that, it often did a very good job when reading the text.

**R Tesseract Results**

The highest accuracy R Tesseract got was 99.2%, which was for image 3028. The lowest accuracy we got was 28.0%, which was for image 3392. Our overall accuracy rate was 65%.

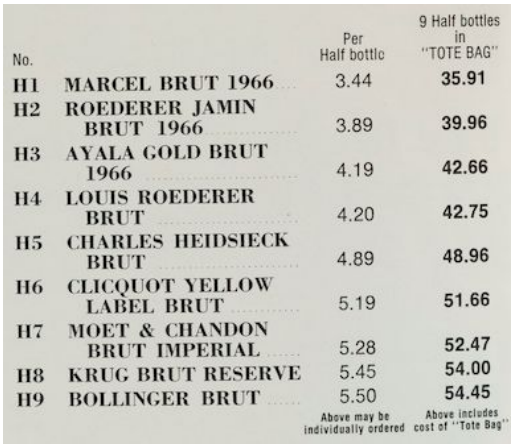
| image | % wrong | % correct |
|---|---|---|
| 0190 | 13.1% | 86.9% |
| 0208 | 17.4% | 82.6% |
| 0237 | 6.9% | 93.1% |
| 0260 | 15.7% | 84.3% |
| 0629 | 5.2% | 94.8% |
| 1198 | 38.9% | 61.1% |
| 2122 | 42.8% | 57.2% |
| 2351 | 21.4% | 78.6% |
| 2362 | 29.7% | 70.3% |
| 2635 | 48.5% | 51.5% |
| 3028 | 0.8% | 99.2% |
| 3392 | 72.0% | 28.0% |
| 3666 | 70.9% | 29.1% |

In some cases, if there were multiple tables in an image, one table may given a high accuracy versus another table in the same image which may have given a low accuracy rate. For example, for image 2122, the highest accuracy was 84.7% for one table in the image, but another table in the exact same image had an accuracy which was as low as 37.6%.

Below, are some sample outputs we got from our model using R Tesseract with the original image for comparison, which was from image 0629 in our dataset:

Parts of output from RTesseract that came out that came out well from image 0629:

| | Name | bottle | case |
|---|---|---|---|
| 1 | MARCEL BRUT 1966 | 3.44 | 35.91 |
| 2 | ROEDERER Jami) BRUT 1966 | 3.89 | 39.96 |
| 3 | AYALA GOLD BRUT 1966 Be | 4.19 | 42.66 |
| 4 | LOUIS ROEDERER BRUT | 4.20 | 42.75 |
| 5 | CHARLES HEIDSIECK BRUT pata | 4.89 | 48.96 |
| 6 | CLICQUOT YELLOW LABEL BRUT | 5.19 | 5.60 |
| 7 | MOET & CHANDON BRUT IMPERIAL | 5.08 | 52.47 |
| 8 | KRUG BRUT RESERVE | 5.45 | 54.00 |
| 9 | BOLLINGER BRUT | 5.50 | 54.45 |

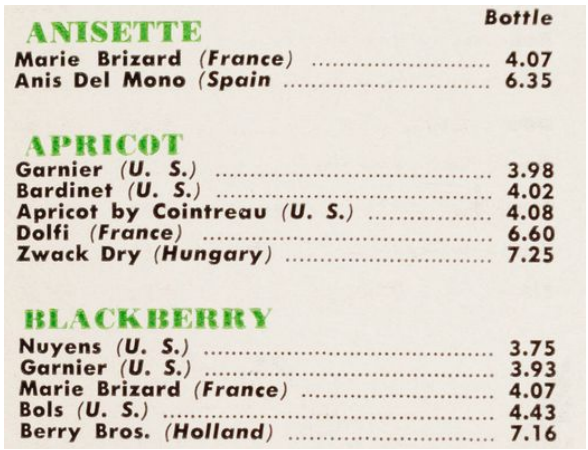


| No. | | Per Half bottle | 9 Half bottles in "TOTE BAG" |
|---|---|---|---|
| H1 | MARCEL BRUT 1966 | 3.44 | 35.91 |
| H2 | ROEDERER JAMIN BRUT 1966 | 3.89 | 39.96 |
| H3 | AYALA GOLD BRUT 1966 | 4.19 | 42.66 |
| H4 | LOUIS ROEDERER BRUT | 4.20 | 42.75 |
| H5 | CHARLES HEIDSIECK BRUT | 4.89 | 48.96 |
| H6 | CLICQUOT YELLOW LABEL BRUT | 5.19 | 51.66 |
| H7 | MOET & CHANDON BRUT IMPERIAL | 5.28 | 52.47 |
| H8 | KRUG BRUT RESERVE | 5.45 | 54.00 |
| H9 | BOLLINGER BRUT | 5.50 | 54.45 |
| | | Above may be individually ordered | Above includes cost of "Tote Bag" |

From the comparison above, the RTesseract output closely mirrors the text in the image. However, it is still not perfect. There were some words that RTesseract got incorrect, and there are words that show up in the output but are not in the image. We calculated that the accuracy for the output is 94.8%. That is, RTesseract got 94.8% of the text correct.

Parts of output from RTesseract that came out poorly from image 2122:

| Name | price1 |
|---|---|
| Marie Brizard (France) | 4.07 |
| Anis Del Mono (Spain | 6.35 |
| Garnier (U 8) coiiiiiiniiib nn | 3.98 |
| BRLAINBI IY 80 os ir tl | 2.02 |
| Apricot by Cointreau (U | S) |
| Doli (France) ii ini | 5.60 |
| Zwack Dry (Hungary) | 7.25 |
| Nuyens | (U |
| Garnier (U 8) coocivsecenserrmmrissrmersssssonnes | 30.93 |
| Marie Brizard (France) | 4.07 |
| Bole tlleiS nr | [pment |
| Berry Bros (Holland) ia | 7.16 |



| | Bottle |
|---|---|
| **ANISETTE** | |
| Marie Brizard (France) | 4.07 |
| Anis Del Mono (Spain | 6.35 |
| **APRICOT** | |
| Garnier (U. S.) | 3.98 |
| Bardinet (U. S.) | 4.02 |
| Apricot by Cointreau (U. S.) | 4.08 |
| Dolfi (France) | 6.60 |
| Zwack Dry (Hungary) | 7.25 |
| **BLACKBERRY** | |
| Nuyens (U. S.) | 3.75 |
| Garnier (U. S.) | 3.93 |
| Marie Brizard (France) | 4.07 |
| Bols (U. S.) | 4.43 |
| Berry Bros. (Holland) | 7.16 |

Although there were certain images where RTesseract extracted the correct names and the prices, there were others where the names and prices did not resemble the text in the image. There were also some images where the output in RTesseract changed the order of the text. Additionally, there were also cases where R Tesseract did not read certain text. When we calculate the accuracy of the RTesseract output, it only had 37.6% accuracy.

**Comparison of Both OCR Methods**

Comparing the outputs of RTesseract and the PDFPenPro, PDFPenpro does not give us the same structure as it is in the image. This means that the order of wines seen in the image is not the same as the output for PDFPenPro. However, the trade off is PDFPenPro gives a higher accuracy than RTesseract in terms of reading the actual names and prices.

## Discussion

**What We Learned**

Before we did this project, none of us were familiar with OCR and the various softwares available. Since the two main tools we used for this project are the RTesseract package and PDFPenPro, we learned a lot about how we could use them to read from images. RTesseract had many parameters that could be changed. Adjusting these would give different results with different kinds of settings. Hence, we tried to adjust different parameters in RTesseract so that we could get the correct wine names and prices, yet it was still not perfect. PDFPenPro was used it differently from Rtesseract. Instead of looking for information line by line, we learned how to extract the whole passage from the text and then dig from the passage to get wine names and prices. Both methods were challenging and it was very interesting to learn about them.

Another thing we learned from this project were some basic machine learning algorithms. At first, we wanted to develop a rudimentary image classification method using deep learning framework using the packages MXNet and Keras in R to help us identify catalog and non-catalog images. However, we realized this was an unnecessary step because we could identify these non-catalog images after we ran OCR on all images. Although in the end we did not use machine learning in this project, we spent some time learning about basic machine learning algorithms and how it could be plausible for our project.

**What We Could Have Improved**

If we had more time, we would have liked to create a function to locate where tables are in the image so that our strategy of extracting information from scanned images would be more completed and general to use in various OCR softwares. This could be very useful and efficient when people need to deal with large number of images. With this function, we could run the rest of the 4,111 images and see how well the OCR methods perform with images that have more complicated formats.

In regards to data analysis, one important fact we learned is that data is not always clean or tidy that are ready to use. For many of the catalog images, the formats and fonts were very different from standard printouts. When we used OCR methods on these images, the results were always random characters which were not readable. In other classes, we normally were given data that was already structured well and we just needed to analyze the data. Therefore, by working on this project, we needed to change our mindset and focus on how we extracted the data that could be used, rather than focusing on the statistical tool we needed to analyze the data, since it was not even readily available for us. Now that we have a method to extract the data and analyze the accuracy of our results, if we had more time doing a statistical analysis of the data could be useful.

Furthermore, we found that meeting each milestone of our project usually took longer than we initially thought. It was very easy to generate ideas on how to solve the problem but it was very difficult to implement. As for working in a group, we could have done better with assigning tasks to each member and making sure that everyone knew exactly what they are responsible for. We also could have had more communication among the group members so that everyone stays on track. In addition, our group would have benefit greatly going to office hours together, and meeting face-to-face more often than we did.

## Conclusion

Our goal for this project was to extract the names and prices of wines and other alcohols from digitized catalogs. While we can say we met that goal under two different methodologies, the ideal situation would use a combination of both of these softwares. The process of doing so would enable the extraction process to have the most accurate results for a majority of the cases. Since there were some big PDFPenPro errors such as failing to recognize any text or tables within certain catalog images where there were actually nicely formatted columns and text, it wouldn’t hurt to preprocess the images in order to provide the software with the best chances to extract the important text. R Tesseract seems to be a more reliable piece of software in the sense that users have more freedom over the configurations as well as how they would implement the functions within the software.