SCI 2000 – Group 8 – Final Report

Erik Huebner, Isham Behl, Muhammad Bilal Sheikh

Professor Max Turgeon

April 16, 2021

Introduction

The initial idea for this analysis began with the question of how the COVID-19 pandemic has financially impacted the world's economy. All decisions thereafter have been guided by this initial, overall research question. After performing an initial analysis on the data collected, this vague research interest is focused into a more specific hypothesis that is then tested with the data collected. As will be explained, the focused research hypothesis is that different industries were affected differently by the pandemic, and that a few key financial metrics can measure the effect of the pandemic on the overall global economy.

In pursuit of answering this question, two sources of data were collected. The first source was financial information on a sampling of large companies. This information was scraped from Yahoo Finance. The companies chosen as a sample were chosen based on an effort to take from a wide variety of countries and industries. First, five industries were selected: Finance/Banking, Tourism/Travel/Airfare, Manufacturing, Food/Agriculture, and Energy/Petroleum. These are hereafter shortened for convenience as Finance, Tourism, Manufacturing, Food, and Energy, respectively. These industries were chosen because it was thought initially that the COVID-19 pandemic would have affected them significantly in some way. Next, 10 companies are chosen from each industry, where each of the 10 companies is based in 10 pre-selected countries. The countries chosen were Canada, United States, Japan, Brazil, Australia, France, Germany, Italy, Singapore, and United Kingdom. Since 5 industries were picked from each of these countries, the result is 50 countries in total being sampled. The actual data collected was two-fold. Firstly, the annual (for the past 3 years) and quarterly (for the past 4 quarters) of income statements were collected for each company. Secondly, the weekly stock price for each company was collected from January 22, 2020 to March 31, 2021.

The second source of data collected was the daily record of confirmed case counts and death totals for the COVID-19 pandemic for each country. The source of the data was the John Hopkins University CSSEGIS data repository for COVID-19 (and can be found at this link: https://github.com/CSSEGISandData/COVID-19). This data was simply downloaded and was not scraped.

Data Collection and Preparation

Data Collection

When attempting to web scrape the financial data from Yahoo Finance, several difficulties were encountered and significant data cleaning and processing was required. To find the full web scraping script, see **Code Appendix A** or the *"SCI 2000 - Financial Data Web Scraper.txt"* file submitted with this report.

The stock price data was accessed easily enough, as the data is stored in an html table class. However, the annual and quarterly financial statements are not. The raw html had to be converted to a string, then regular expressions were used to extract cell elements. Then, loops were used to strong-arm R into interpreting the data correctly and format everything nicely into a table. It is during web scraping that the associated industry, country, and company information is attached to each row in the dataset. The final results of the web scraping (the datasets used at the beginning of the initial analysis and data preparation/cleaning) can be found by accessing, *"annualFinancials.csv"*, *"quarterlyFinancials.csv"*, and *"stockPrices.csv"*, which were all submitted with this report. They are stored as semi-colon deliminated files.

It is relevant to mention that the quarterly income statements in particular were difficult to access. This is because when visiting financial statements on Yahoo Finance, the default setting is annual financial statements. Furthermore, there is no specific link to access quarterly statements; a button must be pressed manually. Therefore, the Docker software along with the RSelenium package was used to achieve the astounding feat of pressing a button (the authors of this paper were humbled by the challenges involved with this simple, button-related task).

It is also crucial to note that this is the first stage where potential sources of error were introduced; not every company had complete financial statements that were reconcilable with the others in the dataset. Although stock prices and annual income statements were able to scrape the data of 49 companies, the quarterly income statements only consist of 33 companies.

The John Hopkins University COVID-19 data was downloaded and saved as *"time_series_covid19_confirmed_global.csv"* and *"time_series_covid19_deaths_global.csv"*. Both were also submitted with this report.

Data Preparation

After data collection, significant further data preparation/cleaning was required to get the data to where it was needed. See **Code Appendix B.2** or the *"SCI 2000 - Data Cleaning and Analysis.txt"* file submitted with this report for the full data cleaning script.

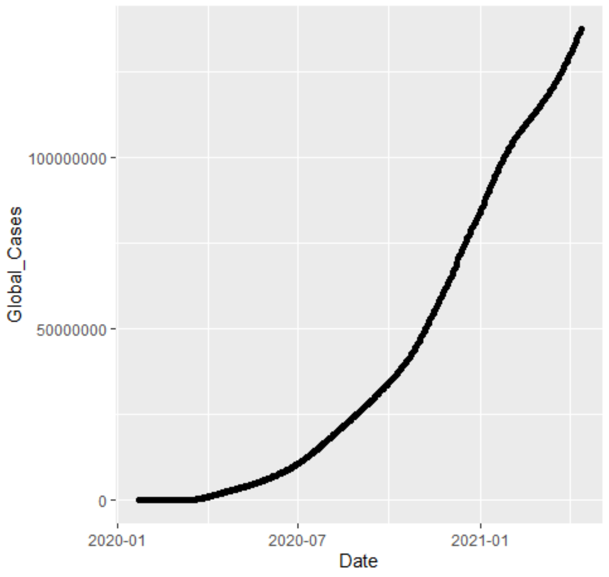
The only work that had to be done to stock prices was adding the country level and global level COVID-19 data; for each company, the global confirmed case count and death count was added along with the same information specific to the country in which the company operated. Additionally, in an effort to normalize stock prices across companies of different size and of different numbers of shares, a new *Return* column was calculated to track the weekly percent change between stock price observations. The final stock price data frame consisted of variables for Date, Closing Price, Percent Return, Country, Industry, Company, Global COVID Deaths, Global COVID Cases, and Cases/Deaths for the specific country. Each observation of this data frame consisted of one week of stock price data and the associated COVID numbers at that time.

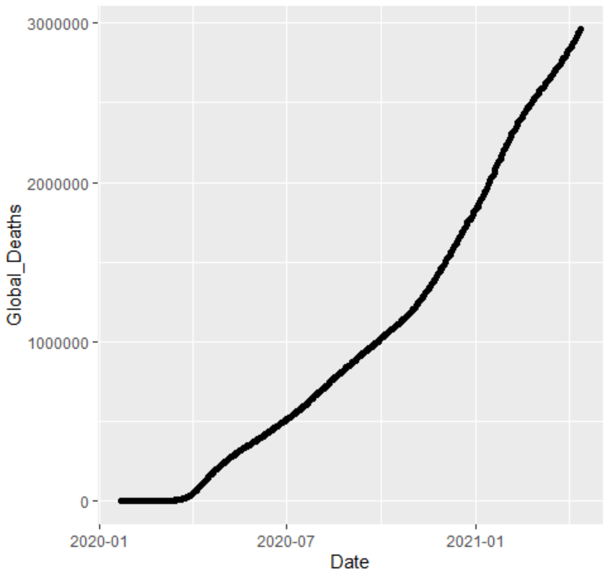
The annual and quarterly income statements required more significant cleaning. Due to the nature of financial statements, the data frames for the income statements had Date values as individual variables, and the class of financial line items (Revenue, Operating Expenses, etc.) as their own, single variable. Therefore, through creative looping, transposing, slicing, and renaming, the data set was transformed so that financial line items were variables, and dates were listed in their own variables as observations in each row. In the transformed data frames, each observation consisted of a particular company's income statements at a particular period in time. The variables of the transformed data frames included Date, Total Revenue, Interest Expense, Income Before Tax, Income Tax Expense, Income from Continuing Operations, Net Income, Net Income Available to Common Shareholders. Variables were also included such as in the stock price data frame such as Country, Industry, and Company, as well as related global/country specific COVID case and death count data.

Finally, a change common to both datasets is currency conversion. Since Yahoo Finance records income statements and stock prices in local currencies, these figures must be converted for any meaningful results to be achieved during analysis. A table was used to attach a currency conversion factor to each row of all three data frames. All currencies were converted to CAD$. This table can be found in the *"conversions.txt"* file submitted with this report.

Initial Analysis

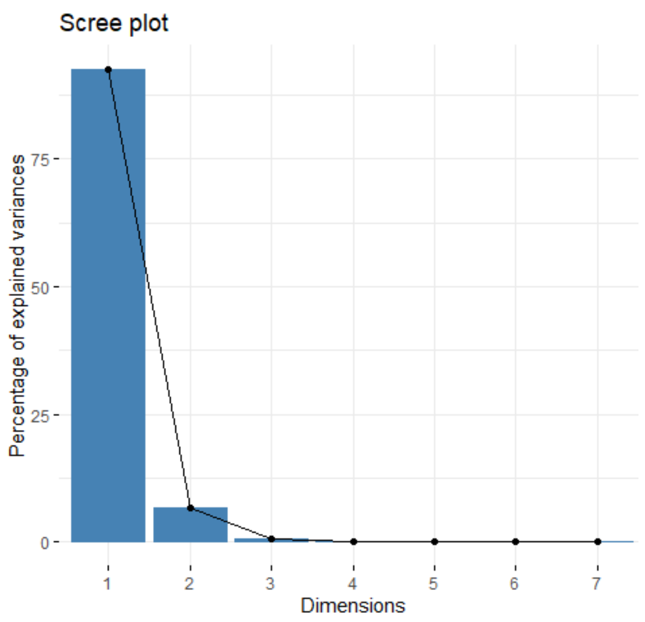
The first step in the exploratory analysis was to plot how confirmed cases and deaths associated with COVID-19 changed over time. The code for the plots below can be found in **Code Appendix C.1**:

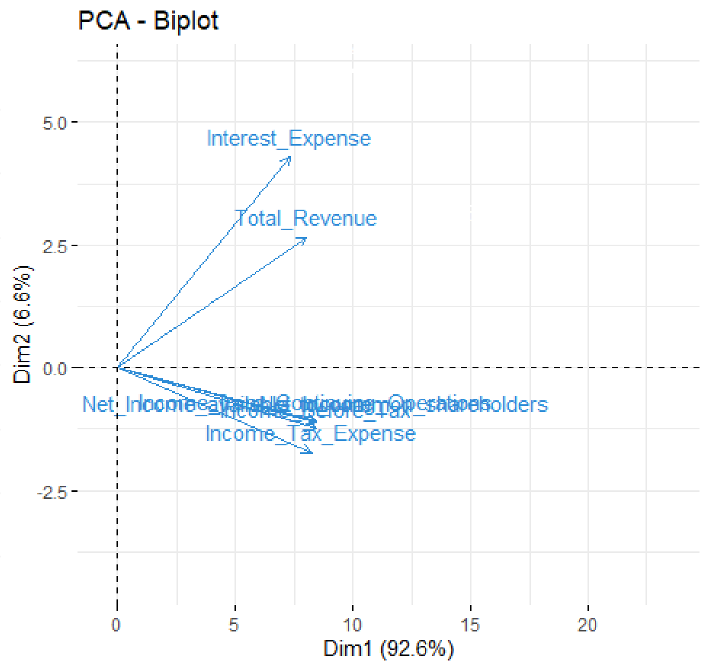




As is obvious to anyone who has not been living under a rock this past year as well as the above graphs, the number of confirmed cases and deaths associated with COVID-19 has not only increased over time, but accelerated as well. One might expect that such a drastic spread of a severe respiratory virus would slow down the global economy. However, this initial hunch must be confirmed with deeper analysis.

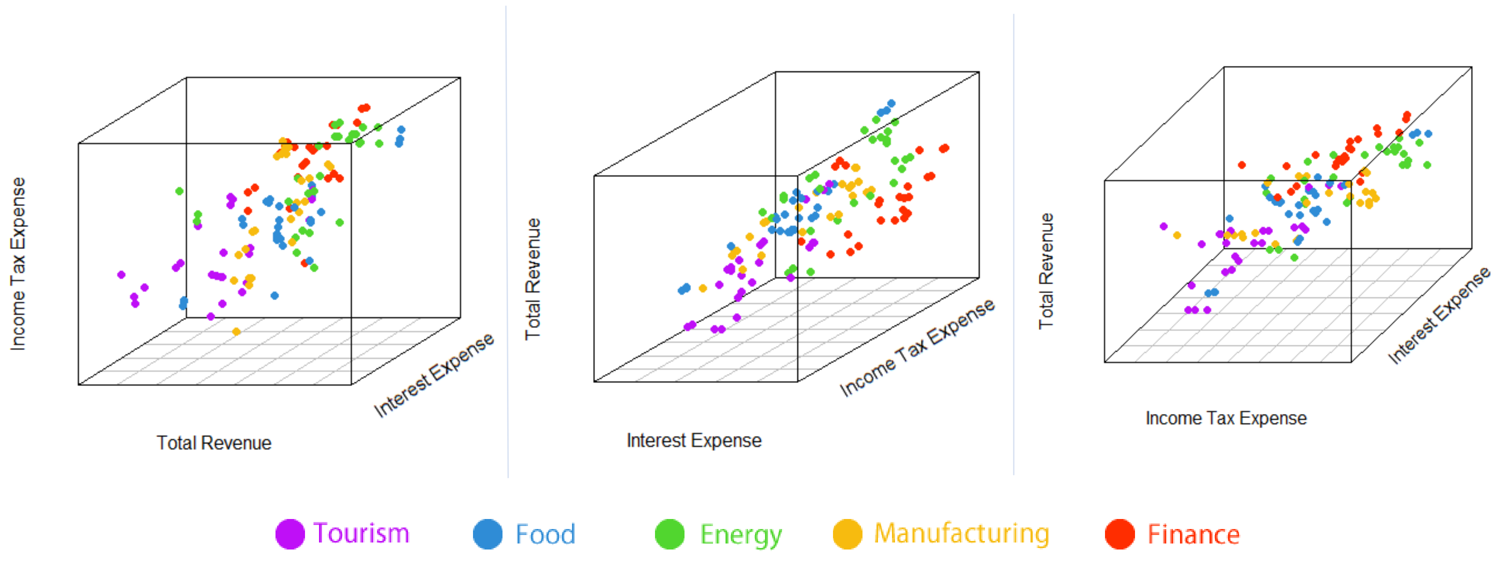
The next step is to confirm the validity of the groupings that have been imposed on the data in terms of industry; the assumption when picking multiple companies from each industry is that companies within an industry will have similar financial data. To confirm this assumption, principal component analysis is performed on the time series annual income statements data frame. To simplify this process, the factoextra library was used. There was two reasons for performing PCA on the income statement data. Firstly, since each financial line item was turned into a variable, 7 financial variables that can be analyzed are present in the data frame. Thus, it would be beneficial to identify a few variables that are the most important and that explain most of the variance in the dataset, so that analysis can be focused. Secondly, it would be beneficial to be able to visualize the data frame along those variables; visualizing the data frame grants the ability to visually confirm that companies in similar industries are clustered similarly. The code for the principal component analysis can be found in **Code Appendix C.2**. The results are shown below:





The above two graphics are a treasure-trove of useful information. The scree plot on the left validates the most important assumption when utilizing PCA: 99.2% of the variance in the data set can be explained by the first two principal components. Then, along these two principal components, the eigenvectors of each variable in the dataset is displayed in the graph on the right. When choosing the variables which are the most important, the key is to look at the absolute value of the components of the eigenvectors along each principal component. As can be seen in the right graph above, most variables have the same component weight along PC1. The main differentiator will therefore be the component along PC2. Using this metric, the most important variables in the data set are Total Revenue, Interest Expense, and Income Tax Expense.

Not only is this conclusion backed up by the above graphics, but also by common sense. Most income statement line items, such as Net Income or Operating Income, are a function of Total Revenue, and would therefore correlate highly with it. Interest Expense, however, is a function of debt, and would be only indirectly connected to Total Revenue. Finally, although Income Tax Expense is partially a function of Total Revenue, the tax rate paid by different companies can vary greatly; some industries, such as airlines or farms, get subsidized heavily, while other industries, such as finance, are naturally more skilled at finding tax loopholes and havens. The dataset is thus graphed according to these three variables, and the results shown below (see **Code Appendix C.3** for these graphs):



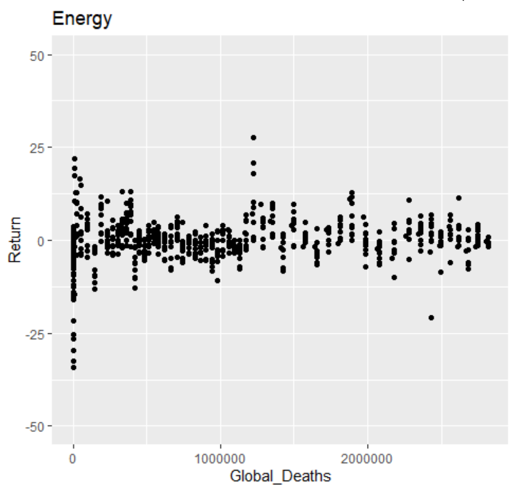
From the above graphs, we can immediately tell that clustering based on industry results in data that is not very linearly separable. However, this is to be somewhat expected, as companies within an economy generally move together. Furthermore, rough, overlapping clusters can still be identified, and industry gives a decently good grouping for the data.

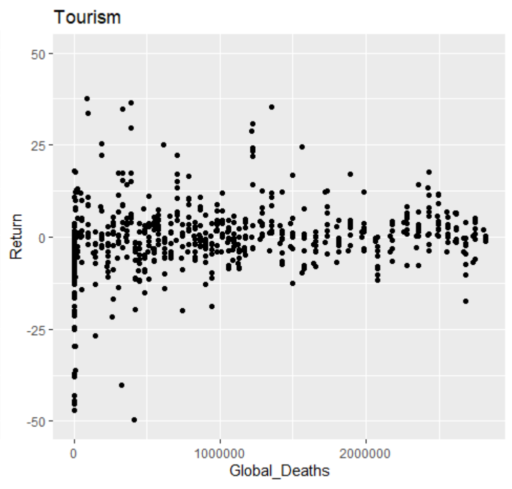
Based on the above principal component analysis and consequent 3D graphing, two conclusions can be drawn. Firstly, the financial data of the 5 industries that were sampled from form a good basis for analysis; different industries performed differently financially and should therefore be examined further to see how COVID-19 impacted each differently. It would be expected that each industry was affected differently. Secondly, the three variables identified as most important, Total Revenue, Interest Expense, and Income Tax Expense should be examined against the rise in COVID-19 cases and deaths, to see how those variables changed as COVID-19 spread.

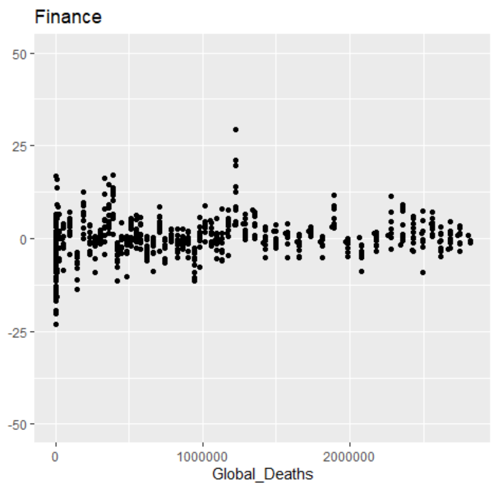
From this initial analysis, our focused research hypothesis can be formed. From the results, a logical hypothesis is that companies in different industries were affected differently by the COVID-19 pandemic and that the affect of the pandemic on the global economy can be measured by Total Revenue, Interest Expense, and Income Tax Expense. Based on preconceived notions (without looking at the data), one would logically hypothesize that the Tourism industry was negatively impacted, while the Food and Manufacturing industries were positively affected and others were neutrally affected. Furthermore, one would logically hypothesize that Total Revenue would decrease as COVID-19 spread (due to people staying at home and not buying as much), Interest Expense would increase (loss of sales would cause companies to take on more debt) and that Income Tax Expense would decrease (stimulus packages might include tax relief covenants and subsidies).

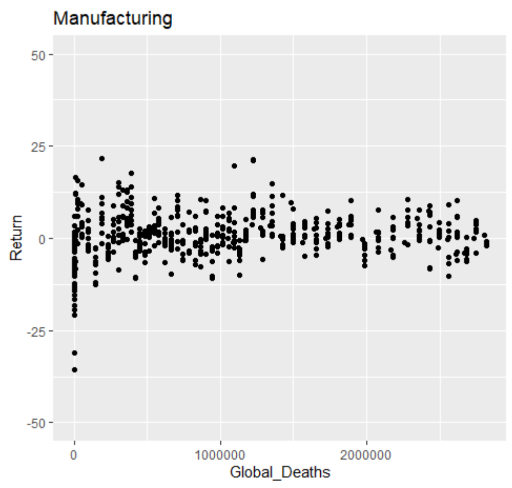
Further Analysis and Discussion

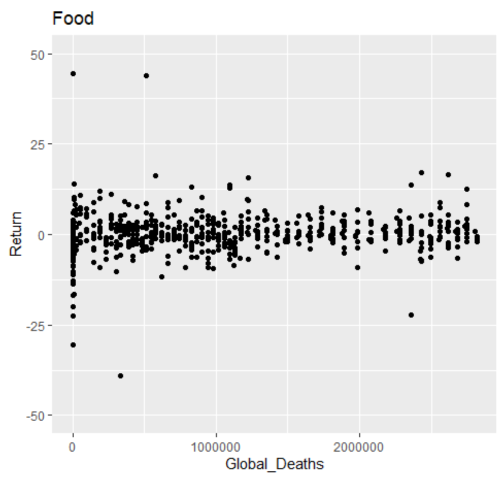
The first step in further analysis was to perform a regression analysis for each industry, measuring the response in stock return for global case counts and death counts of COVID-19. It was found that the results in almost all regression plots that were done were the same when comparing deaths vs. case counts. Therefore, only regressions based on death counts were displayed to save space. Shown below are the scatterplots for each industry of return by global deaths from COVID-19 (see Code Appendix C.4):



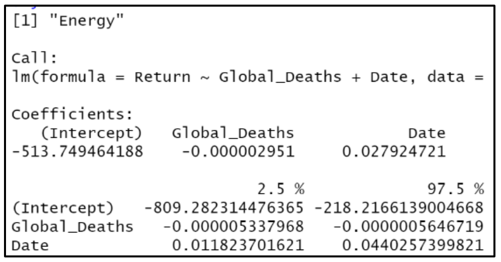




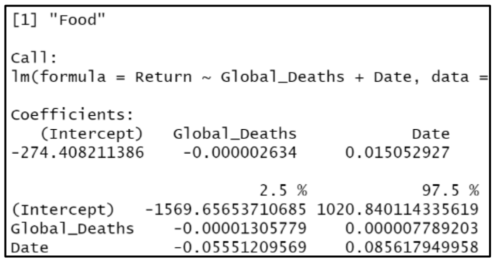




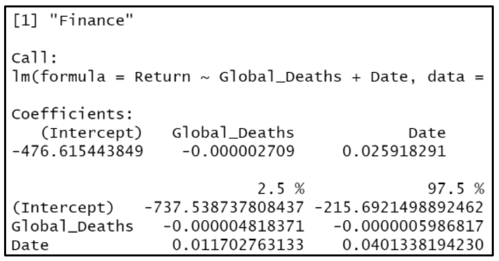
As can be seen in the above graphs, there are far more observations near lower death counts. This is because of the exponential nature of the spread of infections. To account for this, Date is added as a covariate when creating the linear models. However, even from the above graphs the results are promising; the relationships are somewhat linear (albeit slightly sinusoidal). A good next step is to examine the coefficients, their associated confidence intervals, and the residual plots (see Code Appendix C.4):



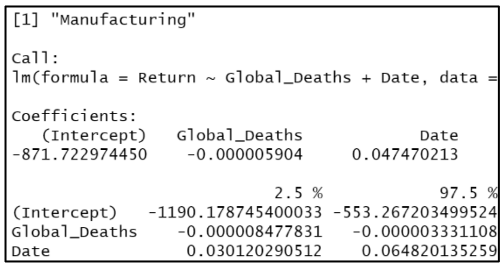
```
[1] "Energy"

Call:
lm(formula = Return ~ Global_Deaths + Date, data =

Coefficients:
   (Intercept)   Global_Deaths            Date
-513.749464188    -0.000002951     0.027924721

                            2.5 %           97.5 %
(Intercept)   -809.282314476365 -218.2166139004668
Global_Deaths   -0.000005337968   -0.0000005646719
Date             0.011823701621    0.0440257399821
```



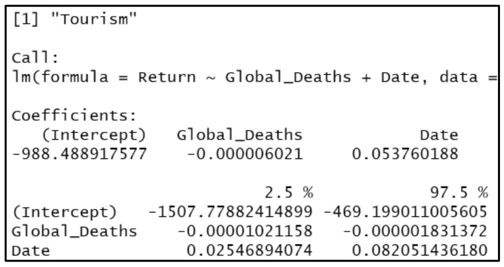
```
[1] "Food"

Call:
lm(formula = Return ~ Global_Deaths + Date, data =

Coefficients:
   (Intercept)   Global_Deaths            Date
-274.408211386    -0.000002634     0.015052927

                            2.5 %          97.5 %
(Intercept)   -1569.65653710685 1020.840114335619
Global_Deaths    -0.00001305779    0.000007789203
Date             -0.05551209569    0.085617949958
```



```
[1] "Finance"

Call:
lm(formula = Return ~ Global_Deaths + Date, data =

Coefficients:
   (Intercept)   Global_Deaths            Date
-476.615443849    -0.000002709     0.025918291

                            2.5 %           97.5 %
(Intercept)   -737.538737808437 -215.6921498892462
Global_Deaths   -0.000004818371   -0.0000005986817
Date             0.011702763133    0.0401338194230
```



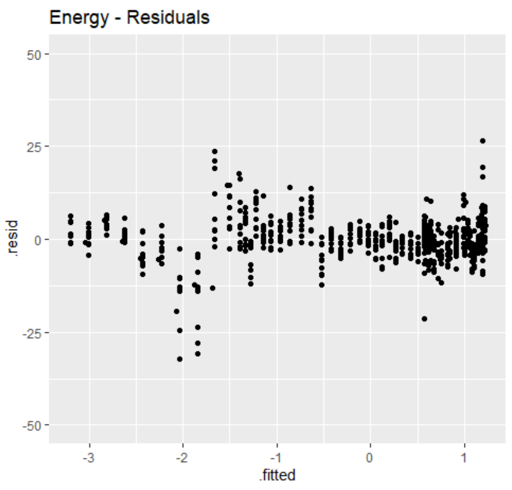
```
[1] "Manufacturing"

Call:
lm(formula = Return ~ Global_Deaths + Date, data =

Coefficients:
   (Intercept)   Global_Deaths            Date
-871.722974450    -0.000005904     0.047470213

                            2.5 %           97.5 %
(Intercept)   -1190.178745400033 -553.267203499524
Global_Deaths    -0.000008477831   -0.000003331108
Date              0.030120290512    0.064820135259
```

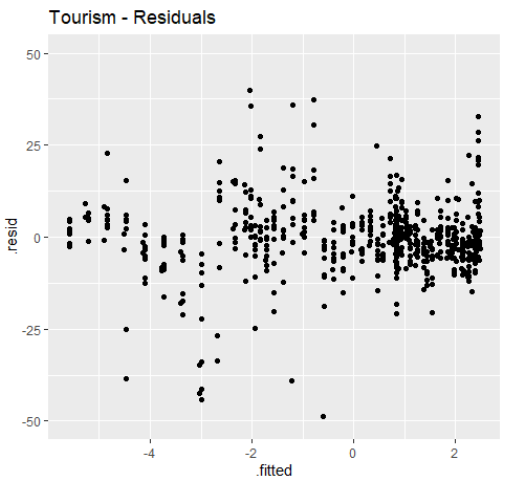


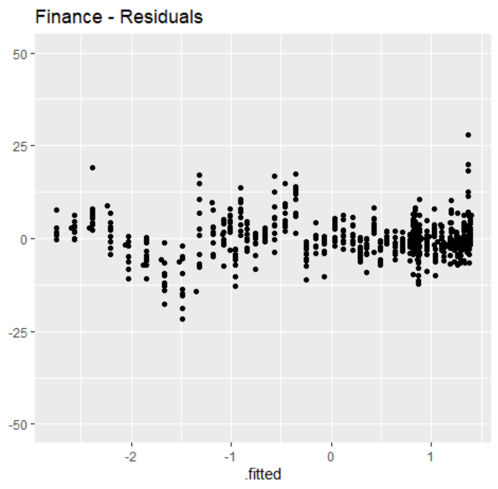
```
[1] "Tourism"

Call:
lm(formula = Return ~ Global_Deaths + Date, data =

Coefficients:
   (Intercept)   Global_Deaths            Date
-988.488917577    -0.000006021     0.053760188

                            2.5 %           97.5 %
(Intercept)   -1507.77882414899 -469.199011005605
Global_Deaths    -0.00001021158   -0.000001831372
Date              0.02546894074    0.082051436180
```

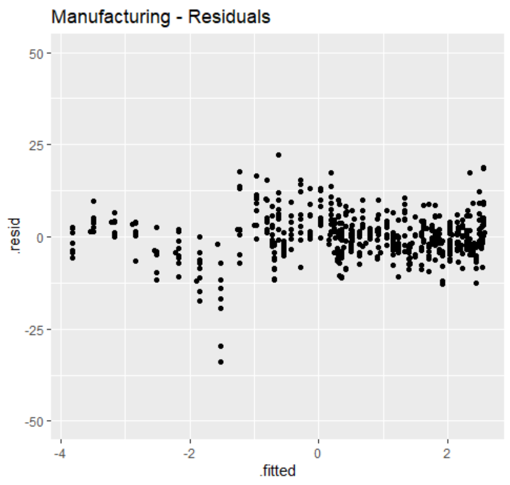
Conclusions can be drawn from the figures above. It is firstly interesting to note that after accounting for date, the regression slope coefficient for all 5 industries is negative when looking at the effect of global deaths from COVID-19. However, the result for all 5 industries is very small. In fact, the result is so small that it is practically negligible. Overall, even though the coefficient is negative, the realistic conclusion is that deaths from COVID-19 had no significant effect on stock returns across all 5 industries. This is not to be confused with saying that the results are statistically insignificant. For Energy, Finance, Manufacturing, and Tourism, the slope coefficients are statistically significant. Therefore, it can be concluded from the evidence that deaths did not affect stock returns. However, the result for Food must be thrown out because the regression coefficient is not statistically significant; the result for the Food industry is that no conclusion can be drawn.

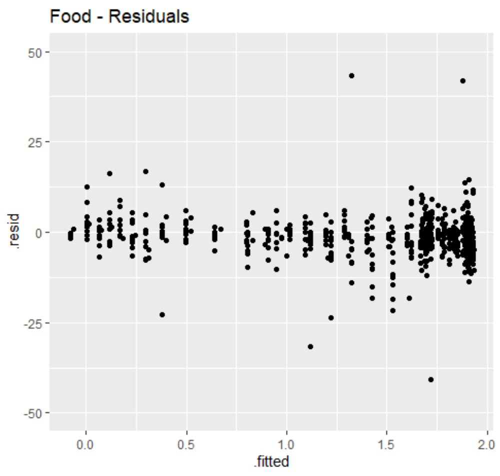
Next, residual plots are shown below to further examine the validity of these regression models:







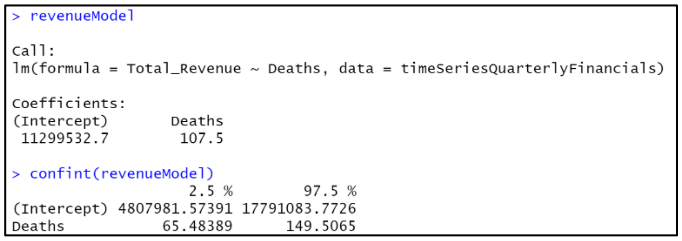




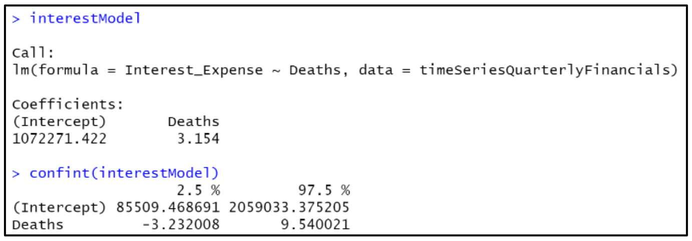
After hours spent on data preparation, the results of the residual analysis are disheartening, to say the least. Looking at the residual plots of the 5 industries, basically all 5 violate both of the top two assumptions for linear models. Firstly, all 5 (with the exception of maybe Food, whose coefficient was statistically insignificant anyway) do have a very curvy relationship to them. That is to say, one would expect to see no relationship present in the graph of fitted versus residual values, whereas in these graphs there is clearly a relationship present. This means that the linearity and additivity assumption is broken. Secondly, all 5 show clear evidence that variance is not constant. This means that homoscedasticity must be violated.

Taking all the evidence together, the correct conclusion is that there is not enough evidence of a relationship between industry stock returns and global deaths from COVID-19. For all 5 industries, either the regression coefficient was not statistically significant, or key assumptions of the linear model were violated. Therefore, the conclusion for the first part of our research hypothesis, that different industries would be affected differently by the pandemic, is that there is no evidence of a differing relationship between industries.

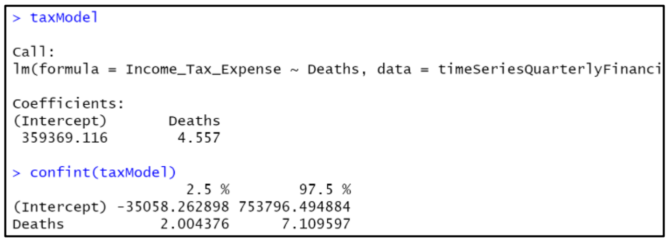
Next, the relationship between global deaths from COVID-19 and key financial variables will be explored (chiefly, Total Revenue, Interest Expense, and Income Tax Expense). This is the second part of our research hypothesis: that the key variables identified during PCA can measuring the effect of COVID-19 on the global economy. First a linear model is created for each variable, measuring the response in that variable from changes in global deaths from COVID-19 (see Code Appendix C.5):



```
> revenueModel

Call:
lm(formula = Total_Revenue ~ Deaths, data = timeSeriesQuarterlyFinancials)

Coefficients:
(Intercept)       Deaths
 11299532.7        107.5

> confint(revenueModel)
                   2.5 %       97.5 %
(Intercept) 4807981.57391 17791083.7726
Deaths           65.48389      149.5065
```

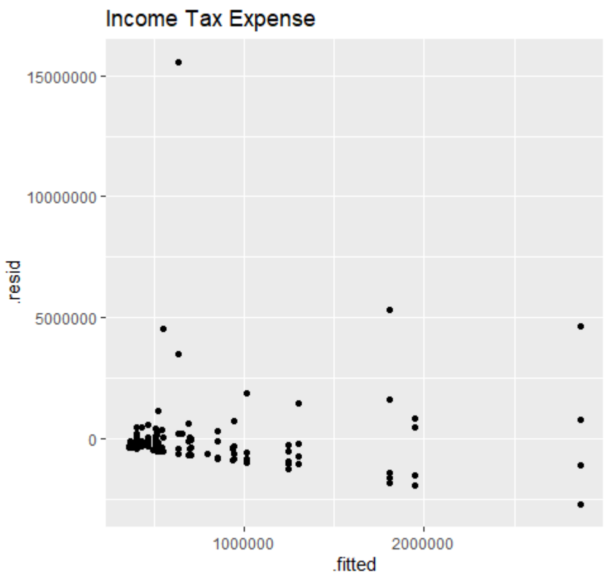


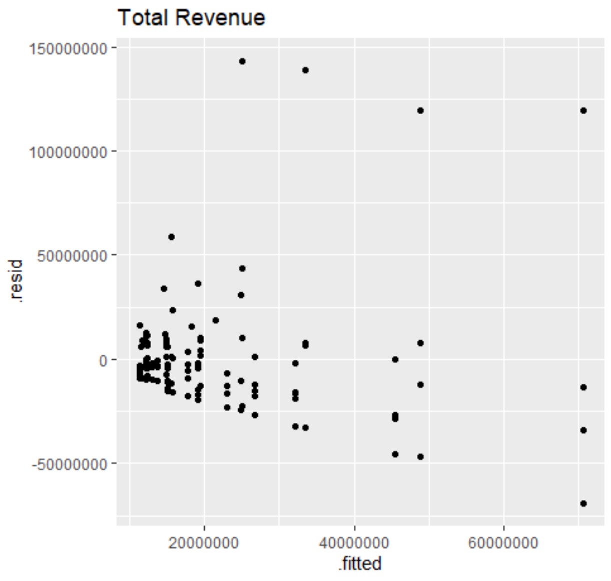
```
> interestModel

Call:
lm(formula = Interest_Expense ~ Deaths, data = timeSeriesQuarterlyFinancials)

Coefficients:
(Intercept)       Deaths
1072271.422        3.154

> confint(interestModel)
                   2.5 %       97.5 %
(Intercept) 85509.468691 2059033.375205
Deaths         -3.232008       9.540021
```



```
> taxModel

Call:
lm(formula = Income_Tax_Expense ~ Deaths, data = timeSeriesQuarterlyFinanci

Coefficients:
(Intercept)       Deaths
 359369.116        4.557

> confint(taxModel)
                   2.5 %       97.5 %
(Intercept) -35058.262898 753796.494884
Deaths          2.004376      7.109597
```

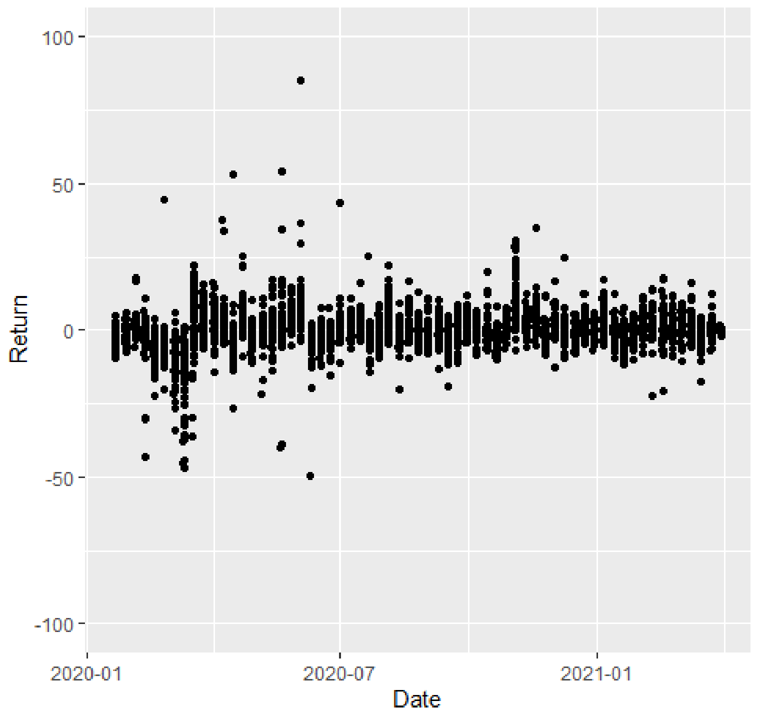
Immediately, several things can be concluded from these figures. Firstly, the slope coefficient for Interest Expense is statistically insignificant, so it will be thrown out immediately and forgotten about. However, the slope coefficients for Total Revenue and Income Tax Expense are statistically significant, and paint a harrowing picture: on average, every new death from COVID-19 was correlated with a $1 07 increase in Total Revenue. This figure becomes even more maddening when you compare it to the average increase in Income Tax Expense per death from COVID-19, which is only $4.50. However, before retaliation is enacted, due diligence must be employed to ensure the results are significant (See Code Appendix C.6):





The graphs above, showing fitted values vs. residuals for Total Revenue and Income Tax Expense, may cause us to put down our pitchforks for a moment. One would expect to see no relationship present in these graphs. However, there is clearly a logarithmic relationship present, meaning that the linearity and additivity assumption is violated. Furthermore, there is clearing no homoscedasticity, meaning that constant variance assumption is also violated. Taking this all into account, the correct conclusion to the second part of our research hypothesis (that Total Revenue, Interest Expense, and Income Tax Expense are good measures for the effect of COVID-19 on the global economy) is that there is insufficient evidence to make a conclusion either way.

In a last pitch effort to make some conclusions with the data that was collected, a simple scatterplot is created showing stock returns for all 5 industries over time (See Code Appendix C.7):



As can be seen in the above graph, stock returns across the economy have stayed relatively stable throughout the entirety of the pandemic. This is in line with most modern economic theories; diversification leads to lower variance of returns. However, it is interesting to note that although case counts and deaths from COVID-19 have been rising steadily throughout the pandemic, stock returns were only really negatively affected around March, when lockdowns and general COVID panic was first occurring.

Therefore, based on this final result, and the results of the analysis, our research hypothesis can be answered. Firstly, based on the evidence which was examined, different industries (Finance, Food, Energy, Tourism, and Manufacturing) were not affected differently by the pandemic. Secondly, key financial metrics (Total Revenue, Interest Expense, Income Tax Return) did not change as a result of rising deaths or case counts and were not able to provide any useful predictive power. Rather, as seen in the last graphic, what really affected the economy the must was the initial onset of panic due to COVID-19. This conclusion is also in line with most modern economic theories; expectation, not reality, is the main driver in economic performance.

Further Considerations and Sources of Error

The biggest source of error that was present in the data set was the small number of companies sampled. Before starting the project, 50 companies seemed like a good enough number. However, once we split them by industry, many of our results became insignificant. If the analysis were to be carried out again, the number of companies sampled would need to be much, much higher. The result of this deficiency was likely noticed throughout the analysis; there was a severe lack of statistically significant linear models and regressions. This problem was exacerbated by the fact that our web scraper was not able to collect all income statements for every company; only about 70% of the initial company list was kept due to incomplete data in the other 30%.

Another consideration is the granularity of the data. Due to the nature of financial statements, new figures are only released every 3 months. This means that even though global death and case counts for COVID-19 would change, the financial figures would not. This undoubtedly would affect the regression analysis in some way. However, unlike the small sample problem, this problem is unavoidable; you cannot force large companies to release financial statements more often.

Code Appendices

Code Appendix A – Web Scraping Script

```
#for this code to work, Docker must be installed and running on the computer

library(data.table)
library(tidyverse)
library(rvest)
library(RSelenium)

getFinancials = function(rawString, htmlData, code, industry, country, companyName, quarter){
  #if quarterly data is collected, get 6 columns, for annual data get 5 columns
  if(quarter){minRows = 6}
  else{minRows = 5}

  allElems = str_extract_all(rawString, "<span data-reactid=\"[0-9]+\">.{1,30}\\s*</span>|data-reactid=\"[0-
9]+\">.{0,5}\\s*</div>|<span>.{1,30}</span|data-test=\"fin-col\">-</div")[[1]]
  allElems = allElems[str_detect(allElems, "\\S+")]
        allSides = html_elements(htmlData, xpath = "//*[@class='Va(m)']")

  headers = html_elements(htmlData, xpath = "//*[@class='D(tbhg)']/div")
  if(!is.na(headers)){
    headers = headers[[1]]

    nCols = str_count(toString(headers), "class=\"D\\(ib\\)|div class=\"Ta\\(c\\)")
    nRows = length(str_extract_all(rawString, "rw-expnded")[[1]])

    currElem = 1
    newDataFrame = data.frame()
    skip = FALSE
    stop = FALSE
  }
  else{
    nCols = -1
  }

  if(nCols < minRows){
    newDataFrame = data.frame()
  }
  else{
    for (i in 1:nRows){
      newRow = c()
      adjust = 0
      skip = FALSE

      if(i > 1){
        newRow[1] = str_replace_all(str_replace_all(str_extract(toString(allSides[i-1]), ">.*<"), "<", ""), ">", "")
        adjust = 1
      }

      if(length(newRow) != 0){
            if(newRow[1] == "Basic EPS"){
               stop = TRUE}
            if(newRow[1] == "Operating Expenses"){
               for(j in 2:nCols){
                  newRow[j] = ""
                  skip = TRUE}
            }
         }

         if(!stop){
            if(!skip){
               for (j in 1:(nCols - adjust)){
                  newElem = str_replace_all(str_replace_all(str_extract(allElems[currElem], ">.*<"), "<", ""), ">", "")
                  currElem = currElem + 1

                  while((!is.na(str_detect(newElem, "\\S+"))) & (!str_detect(newElem, "\\S+"))){
                     newElem = str_replace_all(str_replace_all(str_extract(allElems[currElem], ">.*<"), "<", ""), ">", "")
                     currElem = currElem + 1
                  }

                  newRow[j + adjust] = newElem
               }
            }

            if(length(newDataFrame) == 0){
               newDataFrame = data.frame(newRow)}
            else{
               newDataFrame = cbind(newDataFrame, newRow)}
         }
      }

      #if the most recent data is from 2019, not 2020/21, or if the data was not found, exclude the company from the
dataset
      if((length(newDataFrame) < minRows) | (str_detect(newDataFrame[1,3], "^2019"))){
         newDataFrame = data.frame()
      }
      else{
         newDataFrame = data.table::transpose(newDataFrame)
         names(newDataFrame) = newDataFrame[1,]
         newDataFrame = newDataFrame[2:length(newDataFrame[,1]),]
         newDataFrame = select(newDataFrame, 1:as.integer(minRows))

         newDataFrame = mutate(newDataFrame, Company_Code = code)
         newDataFrame = mutate(newDataFrame, Industry = industry)
         newDataFrame = mutate(newDataFrame, Country = country)
         newDataFrame = mutate(newDataFrame, Company_Name = companyName)
      }
   }

   return(newDataFrame)
}

shell('docker run -d -p 4445:4444 selenium/standalone-firefox')
```

```
driver = remoteDriver(remoteServerAddr = "localhost", port = 4445L, browserName = "firefox")
driver$open()

setwd("D:\\Documents\\University\\new term\\SCI 2000\\Project")
companies = read_delim("companies.txt", delim = ":")

codes = companies$`Company Code`
industries = companies$Industry
countries = companies$Country
companyNames = companies$`Company Name`

init = FALSE
init2 = FALSE

for(i in 1:length(codes)){
	code = codes[i]
	industry = industries[i]
	country = countries[i]
	companyName = companyNames[i]

	url = paste0("https://ca.finance.yahoo.com/quote/", code ,"/financials?p=", code)
	driver$navigate(url)

	#get annual financial statements
	rawReturn = driver$findElement(using = "xpath", value = "//*[@id='Col1-1-Financials-
Proxy']/section/div[3]/div[1]/div")
	rawHtml = rawReturn$getElementAttribute("innerHTML")
	rawString = toString(rawHtml[1])
	htmlData = read_html(rawString)

	if((!init) | (i == 1)){
		annualFinancials = getFinancials(rawString, htmlData, code, industry, country, companyName,
quarter = FALSE)

		#if 2020 data couldnt be found, don't initialize the return data frame
		if(length(annualFinancials) == 0){
			annualFinancials = data.frame()
		}
		else{
			init = TRUE
		}
	}
	else{
		newFrame = getFinancials(rawString, htmlData, code, industry, country, companyName, quarter =
FALSE)

		if(length(newFrame) != 0){
			names(newFrame) = names(annualFinancials)
			annualFinancials = rbind(annualFinancials, newFrame)
		}
	}

	#click until you get to the quarterly reports
	moved = FALSE
	while(!moved){
		button = driver$findElement(using = "xpath", value = "//*[@id='Col1-1-Financials-
Proxy']/section/div[1]/div[2]/button")
		button$clickElement()
		Sys.sleep(0.5)
		button = driver$findElement(using = "xpath", value = "//*[@id='Col1-1-Financials-
Proxy']/section/div[1]/div[2]/button")

		checkString = toString(button$getElementAttribute("innerHTML")[[1]])
		if(str_detect(checkString, "<span>Annual</span>")){
			moved = TRUE
		}
	}

	#get the quarterly financial statements
	rawReturn2 = driver$findElement(using = "xpath", value = "//*[@id='Col1-1-Financials-
Proxy']/section/div[3]/div[1]/div")
	rawHtml2 = rawReturn2$getElementAttribute("innerHTML")
	rawString2 = toString(rawHtml2[1])
	htmlData2 = read_html(rawString2)

	if((!init2) | (i == 1)){
		quarterlyFinancials = getFinancials(rawString2, htmlData2, code, industry, country,
companyName, quarter = TRUE)

		#if 2020/21 data couldnt be found, don't initialize the return data frame
		if(length(quarterlyFinancials) == 0){
			quarterlyFinancials = data.frame()
		}
		else{
			init2 = TRUE
		}
	}
	else{
		newFrame = getFinancials(rawString2, htmlData2, code, industry, country, companyName, quarter
= TRUE)

		if(length(newFrame) != 0){
			names(newFrame) = names(quarterlyFinancials)
			quarterlyFinancials = rbind(quarterlyFinancials, newFrame)
		}
	}

	#get weekly stock prices from Dec 31st, 2019 to March 30, 2021
	url = paste0("https://ca.finance.yahoo.com/quote/", code,
"/history?period1=1577836800&period2=1617148800&interval=1wk&filter=history&frequency=1wk&includeAdjust
edClose=true")
	driver$navigate(url)

	rawReturn = driver$findElement(using = "xpath", value = "//*[@id='Col1-1-HistoricalDataTable-
Proxy']/section/div[2]")
	stockTable = html_table(read_html(rawReturn$getElementAttribute("innerHTML")[[1]]))[[1]]
```

```
		stockTable = mutate(stockTable, Company_Code = code)
    stockTable = mutate(stockTable, Industry = industry)
    stockTable = mutate(stockTable, Country = country)
    stockTable = mutate(stockTable, Company_Name = companyName)

		if(i == 1){
			stockPrices = stockTable
		}
		else{
			stockPrices = rbind(stockPrices, stockTable)
		}

		Sys.sleep(5)
		print(paste0("Finished ", companyName, "."))
}

stockPrices = filter(stockPrices, !str_detect(stockPrices$Date, "^*Close price adjusted for splits."))
write_delim(stockPrices, "D:\\Documents\\University\\new term\\SCI 2000\\Project\\stockPrices.csv", delim = ";")
write_delim(annualFinancials, "D:\\Documents\\University\\new term\\SCI 2000\\Project\\annualFinancials.csv",
delim = ";")
write_delim(quarterlyFinancials, "D:\\Documents\\University\\new term\\SCI
2000\\Project\\quarterlyFinancials.csv", delim = ";")
```

Code Appendix B – Data Preparation

To see the full script, uninterrupted, see "SCI 2000 - Data Cleaning and Analysis.txt" submitted with this report.

#Code Appendix B.1 - Setup

```
setwd("C:\\Users\\Erik\\Documents\\University\\My Classes\\SCI 2000 - Intro to Data Science\\Project")
options(scipen=999)

library(data.table)
library(tidyverse)
library(factoextra)
library(scatterplot3d)
library(broom)
```

#Code Appendix B.2 – Data Preparation/Cleaning

```
############################## Data Preparation ##############################
#read in the files
annualFinancials = read_delim("annualFinancials.csv", ";")
quarterlyFinancials = read_delim("quarterlyFinancials.csv", ";")
stockPrices = read_delim("stockPrices.csv", ";")
casesGlobal = read_csv("time_series_covid19_confirmed_global.csv")
```

```
deathsGlobal = read_csv("time_series_covid19_deaths_global.csv")

#rename variables for convenience
newNames1 = names(annualFinancials)
newNames2 = names(quarterlyFinancials)
newNames3 = names(stockPrices)
newNames1[3:5] = c("2020", "2019", "2018")
newNames2[3:6] = c("Q1_2021", "Q4_2020", "Q3_2020", "Q2_2020")
newNames3[5:6] = c("Close", "Adjusted_Close")
names(annualFinancials) = newNames1
names(quarterlyFinancials) = newNames2
names(stockPrices) = newNames3

#filter out placeholder rows and columns
annualFinancials = filter(annualFinancials, Breakdown != "Operating Expenses")
quarterlyFinancials = filter(quarterlyFinancials, Breakdown != "Operating Expenses")
stockPrices = filter(stockPrices, !str_detect(stockPrices$Open, "Dividend|Stock Split"))
stockPrices = stockPrices[-c(2,3,4,6)] # exclude open, high, low, and adjusted close - all we need is closing price

#replace "-" with 0, removes commas from financials statements, then convert to numeric and convert date strings
to date types
annualFinancials = mutate(annualFinancials, ttm = str_replace_all(ttm, "-", "0"),
  '2020' = str_replace_all(annualFinancials$'2020', "-", "0"),
  '2019' = str_replace_all(annualFinancials$'2019', "-", "0"),
  '2018' = str_replace_all(annualFinancials$'2018', "-", "0"))
annualFinancials = mutate(annualFinancials, ttm = as.numeric(str_replace_all(ttm, ",", "")),
  '2020' = as.numeric(str_replace_all(annualFinancials$'2020', ",", "")),
  '2019' = as.numeric(str_replace_all(annualFinancials$'2019', ",", "")),
  '2018' = as.numeric(str_replace_all(annualFinancials$'2018', ",", "")))
quarterlyFinancials = mutate(quarterlyFinancials, ttm = str_replace_all(ttm, "-", "0"),
  Q1_2021 = str_replace_all(Q1_2021, "-", "0"),
  Q4_2020 = str_replace_all(Q4_2020, "-", "0"),
  Q3_2020 = str_replace_all(Q3_2020, "-", "0"),
  Q2_2020 = str_replace_all(Q2_2020, "-", "0"))
quarterlyFinancials = mutate(quarterlyFinancials, ttm = as.numeric(str_replace_all(ttm, ",", "")),
  Q1_2021 = as.numeric(str_replace_all(Q1_2021, ",", "")),
  Q4_2020 = as.numeric(str_replace_all(Q4_2020, ",", "")),
  Q3_2020 = as.numeric(str_replace_all(Q3_2020, ",", "")),
  Q2_2020 = as.numeric(str_replace_all(Q2_2020, ",", "")))
stockPrices = mutate(stockPrices, Close = as.numeric(str_replace_all(Close, ",", "")))
stockPrices = mutate(stockPrices, Date = str_replace_all(stockPrices$Date, "[,.]", ""))
stockPrices = mutate(stockPrices, Date = as.Date(stockPrices$Date, "%b %d %Y"))

#convert all currencies to CAD
conversionRates = read_delim("conversions.txt", ";")
conversionRates = mutate(conversionRates, Conversion_Rate = as.numeric(Conversion_Rate))

annualFinancials = inner_join(annualFinancials, conversionRates, by = "Company_Code")
quarterlyFinancials = inner_join(quarterlyFinancials, conversionRates, by = "Company_Code")
stockPrices = inner_join(stockPrices, conversionRates, by = "Company_Code")

annualFinancials = mutate(annualFinancials, ttm = ttm/Conversion_Rate, '2020' =
round(annualFinancials$'2020'/Conversion_Rate),
                                              '2019' = round(annualFinancials$'2019'/Conversion_Rate),
                                              '2018' = round(annualFinancials$'2018'/Conversion_Rate))
quarterlyFinancials = mutate(quarterlyFinancials, ttm = ttm/Conversion_Rate, Q1_2021 =
round(Q1_2021/Conversion_Rate),
                                              Q4_2020 = round(Q4_2020/Conversion_Rate),
                                              Q3_2020 = round(Q3_2020/Conversion_Rate),
                                              Q2_2020 = round(Q2_2020/Conversion_Rate))
stockPrices = mutate(stockPrices, Close = Close/Conversion_Rate)

#create the return column to normalize stock prices across companies
i = 1
for(company in c(levels(factor(stockPrices$Company_Code)))){
  tempStocks = filter(stockPrices, Company_Code == company)

  temp = tempStocks$Close
  temp = temp[2:length(temp)]
  temp = c(temp, 0)
  tempStocks = mutate(tempStocks, Return = ((Close - temp)/(temp))*100)
  tempStocks = tempStocks[1:(nrow(tempStocks) - 1),]

  if(i == 1){newStocks = tempStocks}
  else{newStocks = rbind(newStocks, tempStocks)}
  i = i + 1
}
stockPrices = newStocks

#turn the quarterly financial data into a time series dataset
#since different data is available for different companies, keep only common line items.
keepCols = c("Total Revenue", "Interest Expense", "Income Before Tax", "Income Tax Expense", "Income from
Continuing Operations", "Net Income", "Net Income available to common shareholders")

i = 1
for(companyCode in c(levels(factor(quarterlyFinancials$Company_Code)))){
  temp = filter(quarterlyFinancials, Company_Code == companyCode & Breakdown %in% keepCols)

  if(nrow(temp) == 7){
    companyName = toString(temp[1,10])
    industry = toString(temp[1,8])
    country = toString(temp[1,9])

    for(col in 3:6){
      dateVal = toString(names(temp)[col])
      tempRow = temp[,c(1,col)]

      tempRow = data.table::transpose(tempRow)
      tempRow = cbind(data.frame(c("Date", dateVal)), tempRow, data.frame(c("Country", country)),
data.frame(c("Industry", industry)), data.frame(c("Company_Name", companyName)),
data.frame(c("Company_Code", companyCode)))
      names(tempRow) = tempRow[1,]
      tempRow = tempRow[2,]
      row.names(tempRow) = 1

      if(i == 1){newDataFrame = tempRow}
      else{newDataFrame = rbind(newDataFrame, tempRow)}
      i = i + 1
    }
  }
}
row.names(newDataFrame) = c(1:nrow(newDataFrame))
for(col in 2:8){newDataFrame[,col] = as.numeric(newDataFrame[,col])}
names(newDataFrame) = str_replace_all(names(newDataFrame), " ", "_")
timeSeriesQuarterlyFinancials = newDataFrame

#turn the annual financial data into a time series dataset
i = 1
for(companyCode in c(levels(factor(annualFinancials$Company_Code)))){
  temp = filter(annualFinancials, Company_Code == companyCode & Breakdown %in% keepCols)

  if(nrow(temp) == 7){
    companyName = toString(temp[1,9])
    industry = toString(temp[1,7])
    country = toString(temp[1,8])

    for(col in 3:5){
      dateVal = toString(names(temp)[col])
      tempRow = temp[,c(1,col)]

      tempRow = data.table::transpose(tempRow)
      tempRow = cbind(data.frame(c("Date", dateVal)), tempRow, data.frame(c("Country", country)),
data.frame(c("Industry", industry)), data.frame(c("Company_Name", companyName)),
data.frame(c("Company_Code", companyCode)))
      names(tempRow) = tempRow[1,]
      tempRow = tempRow[2,]
      row.names(tempRow) = 1

      if(i == 1){newDataFrame = tempRow}
      else{newDataFrame = rbind(newDataFrame, tempRow)}
      i = i + 1
    }
  }
}
row.names(newDataFrame) = c(1:nrow(newDataFrame))
for(col in 2:8){newDataFrame[,col] = as.numeric(newDataFrame[,col])}
names(newDataFrame) = str_replace_all(names(newDataFrame), " ", "_")
timeSeriesAnnualFinancials = newDataFrame

#convert quarter labels to date types for quarterly time series
for(i in 1:nrow(timeSeriesQuarterlyFinancials)){
  if(timeSeriesQuarterlyFinancials[i,1] == "Q1_2021"){timeSeriesQuarterlyFinancials[i,1] = "2021-03-30"}
  else if (timeSeriesQuarterlyFinancials[i,1] == "Q4_2020"){timeSeriesQuarterlyFinancials[i,1] = "2020-12-30"}
  else if (timeSeriesQuarterlyFinancials[i,1] == "Q3_2020"){timeSeriesQuarterlyFinancials[i,1] = "2020-09-30"}
  else {timeSeriesQuarterlyFinancials[i,1] = "2020-06-30"}}
timeSeriesQuarterlyFinancials[,1] = as.Date(timeSeriesQuarterlyFinancials[,1])

#convert annual labels to date types for annual time series
for(i in 1:nrow(timeSeriesAnnualFinancials)){
  if(timeSeriesAnnualFinancials[i,1] == "2020"){timeSeriesAnnualFinancials[i,1] = "2020-12-31"}
  else if (timeSeriesAnnualFinancials[i,1] == "2019"){timeSeriesAnnualFinancials[i,1] = "2019-12-31"}
  else {timeSeriesAnnualFinancials[i,1] = "2018-12-31"}}
timeSeriesAnnualFinancials[,1] = as.Date(timeSeriesAnnualFinancials[,1])


############################################## Preparing Covid Data
##############################################

#remove province, latitute and longitude, and rename country variable
casesGlobal = select(casesGlobal, -c(1,3,4))
deathsGlobal = select(deathsGlobal, -c(1,3,4))
temp = names(casesGlobal)
temp[1] = "Country"
names(casesGlobal) = temp
temp = names(deathsGlobal)
temp[1] = "Country"
names(deathsGlobal) = temp

#change US to United States
casesGlobal = mutate(casesGlobal, Country = if_else(Country == "US", "United States", Country))
deathsGlobal = mutate(deathsGlobal, Country = if_else(Country == "US", "United States", Country))

#define a function which groups covid data by country so there is one row per country
groupByCountry = function(dataset){
  i = 1
  while(i <= nrow(dataset)){
    check = dataset[i,1]
    allRows = filter(dataset, Country == toString(check))
    adjust = nrow(allRows)
    allNums = allRows[,-c(1)]
    newRow = cbind(check, data.table::transpose(data.frame(colSums(allNums))))
    names(newRow) = names(dataset)

    if(i == 1){newDataFrame = newRow}
    else{newDataFrame = rbind(newDataFrame, newRow)}
    i = i + adjust
  }

  return(newDataFrame)
}

casesGlobal = groupByCountry(casesGlobal)
deathsGlobal = groupByCountry(deathsGlobal)

#takes a covid dataset and turns it into a time-series based one
makeTimeSeries = function(dataset){
  #transpose the data to turn dates into values and countries into variables
  newDataFrame = data.table::transpose(dataset)

  #make the country values into column headers, then remove that row
  row.names(newDataFrame) = names(dataset)
  newDataFrame = cbind(row.names(newDataFrame), newDataFrame)
```

```
    names(newDataFrame) = newDataFrame[1,]
    newDataFrame = newDataFrame[2:nrow(newDataFrame),]
    newColNames = names(newDataFrame)
    newColNames[1] = "Date"
    names(newDataFrame) = newColNames
    row.names(newDataFrame) = c(1:nrow(newDataFrame))

    return(newDataFrame)
}

timeSeriesCases = makeTimeSeries(casesGlobal)
timeSeriesDeaths = makeTimeSeries(deathsGlobal)

#convert string Dates to date types
timeSeriesCases = mutate(timeSeriesCases, Date = as.Date(timeSeriesCases$Date, "%m/%d/%y"))
timeSeriesDeaths = mutate(timeSeriesDeaths, Date = as.Date(timeSeriesCases$Date, "%m/%d/%y"))

#convert all non-date columns to numerics
for(i in 2:ncol(timeSeriesCases)){
    timeSeriesCases[,i] = as.numeric(timeSeriesCases[,i])}
for(i in 2:ncol(timeSeriesDeaths)){
    timeSeriesDeaths[,i] = as.numeric(timeSeriesDeaths[,i])}

#define a function which adds country based and worldwide covid totals to each row of a time-series dataset
addCovidData = function(dataset, timeCases, timeDeaths){
    #tally global deaths over time
    allDeaths = select(timeDeaths, c(1,2))
    allDeaths[,2] = rowSums(timeDeaths[,2:ncol(timeDeaths)])
    names(allDeaths) = c("Date", "Global_Deaths")

    #tally global cases over time
    allCases = select(timeCases, c(1,2))
    allCases[,2] = rowSums(timeCases[,2:ncol(timeCases)])
    names(allCases) = c("Date", "Global_Cases")

    #loop through each country to calculate totals
    i = 1
    for(country in c(levels(factor(dataset$Country)))){
        #get all rows from the dataset corresponding to that country
        tempRows = filter(dataset, Country == country)

        #join cases and deaths on date
        tempCases = select(timeCases, c('Date', country))
        tempDeaths = select(timeDeaths, c('Date', country))
        names(tempCases) = c("Date", "Cases")
        names(tempDeaths) = c("Date", "Deaths")

        tempCounts = inner_join(tempCases, tempDeaths, by = "Date")
        names(tempCounts) = c("Date", "Cases", "Deaths")
        joined = inner_join(tempRows, tempCounts, by = "Date")
        joined = inner_join(joined, allCases, by = "Date")
        joined = inner_join(joined, allDeaths, by = "Date")
        if(i == 1){newDataset = joined}
        else{newDataset = rbind(newDataset, joined)}
        i = i + 1
    }

    newDataset = mutate(newDataset, Cases = as.numeric(Cases))
    newDataset = mutate(newDataset, Deaths = as.numeric(Deaths))
    newDataset = mutate(newDataset, Global_Deaths = as.numeric(Global_Deaths))
    newDataset = mutate(newDataset, Global_Cases = as.numeric(Global_Cases))

    return(newDataset)
}

#add the covid data to time series annual financial statements and stock prices
stockPrices = addCovidData(stockPrices, timeSeriesCases, timeSeriesDeaths)
timeSeriesQuarterlyFinancials = addCovidData(timeSeriesQuarterlyFinancials, timeSeriesCases, timeSeriesDeaths)
```

#Code Appendix C.1. – Initial Time Series Plots

```
#tally global deaths over time
allDeaths = select(timeSeriesDeaths, c(1,2))
allDeaths[,2] = rowSums(timeSeriesDeaths[,2:ncol(timeSeriesDeaths)])
names(allDeaths) = c("Date", "Global_Deaths")

#tally global cases over time
allCases = select(timeSeriesCases, c(1,2))
allCases[,2] = rowSums(timeSeriesCases[,2:ncol(timeSeriesCases)])
names(allCases) = c("Date", "Global_Cases")

#plot cases and deaths over time
ggplot(data = allDeaths, aes(x=Date,y=Global_Deaths)) + geom_point()
ggplot(data = allCases, aes(x=Date,y=Global_Cases)) + geom_point()
```

#Code Appendix C.2. – Principal Component Analysis

```
#do the principal component analysis
data = timeSeriesAnnualFinancials[,2:8] #take only numerical data
pca = prcomp(data, scale = TRUE) #run the pca function
fviz_eig(pca) #visualize variance explainable by principal components
fviz_pca_biplot(pca, col.var = "#2F8CD6", col.ind = "#FFFFFF") #visualize principal components
```

#Code Appendix C.3. – 3D Scatterplot

```
#data = filter(timeSeriesAnnualFinancials, Company_Code != "MUFG" & Company_Code != "ITUB" & Company_Code
!= "JPM" & Company_Code != "HSBC")
data = filter(timeSeriesAnnualFinancials, Company_Code != "MUFG" & Company_Code != "ITUB" & Company_Code
!= "LTM.SN" & Company_Code !=  "500.SI")
```

```
data = select(data, c("Total_Revenue", "Interest_Expense", "Income_Tax_Expense", "Industry"))

#create a color list
colorPallate = c("#FF2D00", "#51D62F", "#2F8CD6", "#F7BB0F", "#BF0FF7")
industries = c("Finance", "Energy", "Food", "Manufacturing", "Tourism")
colors = rep("", nrow(data))
for(i in 1:nrow(data)){colors[i] = colorPallate[match(data[i,4], industries)]}

#take the log of the selected variables
logInterest = log(data$Interest_Expense)
logRevenue = log(data$Total_Revenue)
logTax = log(data$Income_Tax_Expense)

#graph the selected log variables
scatterplot3d(x = logRevenue, y = logInterest, z = logTax, color = colors, tick.marks = FALSE, pch = 16, xlab = "Total
Revenue", ylab = "Interest Expense", zlab = "Income Tax Expense")
scatterplot3d(x = logInterest, y = logTax, z = logRevenue, color = colors, tick.marks = FALSE, pch = 16, xlab = "Interest
Expense", ylab = "Income Tax Expense", zlab = "Total Revenue")
scatterplot3d(x = logTax, y = logRevenue, z = logInterest, color = colors, tick.marks = FALSE, pch = 16, xlab = "Income
Tax Expense", ylab = "Interest Expense", zlab = "Total Revenue")
```

#Code Appendix C.4. – Industry Scatterplots/Linear Models

```
for(industry in levels(factor(stockPrices$Industry))){
  data = filter(stockPrices, Industry == industry)
  model = lm(Return ~ Global_Deaths + Date, data = data)
  print(industry)
  print(model)
  print(confint(model))
  print((ggplot(data = data, aes(x = Global_Deaths, y = Return)) + geom_point() + ylim(-50,50) +
ggtitle(toString(industry))))
  print((ggplot(data = augment(model), aes(x = .fitted, y = .resid)) + geom_point() +
ggtitle(paste0(toString(industry), " - Residuals")) + ylim(-50,50)))
}
```

#Code Appendix C.5. – Financial Metric Linear Models

```
revenueModel = lm(Total_Revenue ~ Deaths, data = timeSeriesQuarterlyFinancials)
interestModel = lm(Interest_Expense ~ Deaths, data = timeSeriesQuarterlyFinancials)
taxModel = lm(Income_Tax_Expense ~ Deaths, data = timeSeriesQuarterlyFinancials)

revenueModel
confint(revenueModel)
interestModel
confint(interestModel)
taxModel
confint(taxModel)
```

#Code Appendix C.6. – Financial Metric Residual Analysis

```
ggplot(data = augment(revenueModel), aes(x=.fitted,y=.resid)) + geom_point() + ggtitle("Total Revenue")
ggplot(data = augment(taxModel), aes(x=.fitted,y=.resid)) + geom_point() + ggtitle("Income Tax Expense")
```

#Code Appendix C.7. – Death Counts Time Series

```
ggplot(data = stockPrices, aes(x=Date, y=Return)) + geom_point() + ylim(-100,100)
```