

Reinforcing Data Extraction, Transformation, and Loading Skills Using R

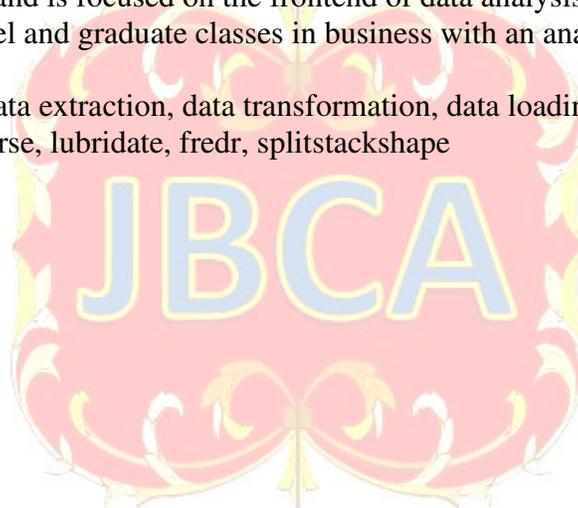
Dwayne Powell
Arkansas State University

Robert Knisley
Wheaton College

ABSTRACT

“Real-world data” is often messy and includes unneeded information. Most textbook datasets are small, clean, easily managed and the exercises focus on analysis and decision-making (backend of data analysis). Using publicly available data from three different sources, this case study (project) requires students to import large datasets and clean and merge the data contained in the datasets and is focused on the frontend of data analysis. The case/project is appropriate for upper-level and graduate classes in business with an analytics component.

Keywords: R, RStudio, data extraction, data transformation, data loading, ETL, data cleaning, data manipulation, tidyverse, lubridate, fredr, splitstackshape



Copyright statement: Authors retain the copyright to the manuscripts published in AABRI journals. Please see the AABRI Copyright Policy at <http://www.aabri.com/copyright.html>

CASE STUDY

1.1 Introduction

Technology continues to change at an unprecedented pace and the skills required for data analysis and decision-making are evolving (AICPA, 2020) requiring accountants to have a specialized skillset (i.e., those of a data scientist) to extract meaningful information from large data sets (Raschke & Charron, 2021). Consequently, the American Institute of Certified Public Accountants (AICPA), the National Association of State Boards of Accountancy (NASBA), and the American Association of Accountants (AAA) agree that accountants and CPA exam candidates need technical data analytic skills. As a result, the current iteration of the CPA exam includes increased testing of data analytic skills in the Business Environment and Concepts (BEC) and Auditing and Attestation (AUD) sections.

“Real-world data” is often messy and includes unneeded information that needs to be cleaned and filtered. According to an article in Forbes magazine, approximately 80% of data scientists’ time is spent extracting and preparing data (Press, 2016). This case study (project) aims to enhance the data science skills needed by today’s accountants. In this project, you are asked to import large datasets, clean, and combine the data in preparation for analysis and decision making. This is an ETL (extract, transform, and load) project that uses R software (R Core Team, 2022) and datasets available from public sources to reinforce data extraction and preparation skills.

1.2 Learning Objectives.

After completing this case, you will be able to use R to:

- Import large datasets.
- Identify specific variables using a data dictionary and other sources.
- Select specific variables.
- Combine multiple datasets.
- Create calculated variables.
- Filter out observations based on specified criteria.

While this case is focused on Extraction, Transformation, and Loading (ETL), your instructor may choose to ask you to analyze the data and demonstrate an ability to reach critical conclusions.

1.3 Data

This case utilizes data from three different public sources: bank holding company financial reports¹, economic data from the St. Louis Federal Reserve (FRED), and regulation data from QuantGov.com. You will import and combine the bank holding company reports for 2011 – 2021. You will be provided with links to download the dataset(s) of all bank holding

¹ Bank knowledge is not required to complete this case. The tools utilized in this case may be applied to any dataset. Bank data was selected because of its public availability and convenience.

company financial reports from the Chicago Federal Reserve, economic data from the St. Louis Federal Reserve (FRED)², and regulatory restrictions from QuantGov.com.

1.4 Software

This case uses R, a free, open-source statistical software package. R can be downloaded from <https://www.r-project.org/> and is available for Windows, UNIX, and macOS platforms. Additionally, we recommend using RStudio, an integrated development environment (IDE) for R. An IDE makes writing, running, and debugging your scripts more user-friendly. RStudio can be downloaded from <https://www.rstudio.com/products/rstudio/> and is also available for Windows, UNIX, and macOS platforms. You will be asked to install and load various R packages throughout the case, including Tidyverse (<https://www.tidyverse.org>), Lubridate (<https://lubridate.tidyverse.org>), and fredr, among others. Tidyverse is a collection of R packages designed for data science. Lubridate is a package that makes it easier to work with dates than the R base package. Fredr allows users to search for and import time-series observations and associated metadata within the FRED database (*Fredr*, n.d.). It is only necessary to install packages once; however, each package must be loaded each time you start a new session. There are many free R resources available online. A simple search will often yield hundreds or thousands of hits. R for Data Science (free online) is helpful <https://r4ds.had.co.nz/>. This case was written using R version 4.2.0, and RStudio 2022.02.2 Build 485.

1.5 Prerequisites

To complete this case, you should have a basic understanding of R and RStudio, a computer with R and RStudio installed or the ability to install R and RStudio, and a high-speed internet connection.

1.6 Requirements

1. If R and/or RStudio are not installed on your computer, download and install R and RStudio.
 - a. Download R from <https://www.r-project.org/>.
 - i. Follow onscreen instructions.
 - b. Download RStudio from <https://www.rstudio.com/products/rstudio/>.
 - i. Follow onscreen instructions.
2. Install the Tidyverse Package
3. Install the Lubridate Package
4. Install the Fredr Package
5. Install the Splitstackshape Package
6. Load the Tidyverse, Lubridate, and Fredr packages.

² To import FRED data, you will need an account and API key, which can be obtained from https://fred.stlouisfed.org/docs/api/api_key.html. Accounts and API keys are free for educational and research purposes. These datasets were selected for this case because they are publicly available and are large enough to present basic challenges during the ETL processes.

7. Download the 2011 bank holding company financial reports and import the dataset into R using the following link <https://www.chicagofed.org/~media/others/banking/financial-institution-reports/bhc-data/bhcf1112.csv>. Create a data frame named BHCF_2011.
8. Download the 2012 bank holding company financial reports and import the dataset into R using the following link <https://www.chicagofed.org/~media/others/banking/financial-institution-reports/bhc-data/bhcf1212.csv>. Create a data frame named BHCF_2012.
9. Download the 2013 bank holding company financial reports and import the dataset into R using the following link <https://www.chicagofed.org/~media/others/banking/financial-institution-reports/bhc-data/bhcf1312.csv>. Create a data frame named BHCF_2013.
10. Download the 2014 bank holding company financial reports and import the dataset into R using the following link <https://www.chicagofed.org/~media/others/banking/financial-institution-reports/bhc-data/bhcf1412.csv>. Create a data frame named BHCF_2014.
11. Download the 2015 bank holding company financial reports and import the dataset into R using the following link <https://www.chicagofed.org/~media/others/banking/financial-institution-reports/bhc-data/bhcf1512.csv>. Create a data frame named BHCF_2015.
12. Download the 2016 bank holding company reports and import the dataset into R using the following link <https://www.chicagofed.org/~media/others/banking/financial-institution-reports/bhc-data/bhcf1612.csv>. Name the dataset BHCF_2016.
13. Download the 2017 bank holding company reports and import the dataset into R using the following link <https://www.chicagofed.org/~media/others/banking/financial-institution-reports/bhc-data/bhcf1712.csv>. Name the dataset BHCF_2017.
14. Download the 2018 bank holding company reports and import the dataset into R using the following link <https://www.chicagofed.org/~media/others/banking/financial-institution-reports/bhc-data/bhcf1812.csv>. Name the dataset BHCF_2018.
15. Download the 2019 bank holding company reports and import the dataset into R using the following link <https://www.chicagofed.org/~media/others/banking/financial-institution-reports/bhc-data/bhcf1912.csv>. Name the dataset BHCF_2019.
16. Download the 2020 bank holding company reports and import the dataset into R using the following link <https://www.chicagofed.org/~media/others/banking/financial-institution-reports/bhc-data/bhcf2012.csv>. Name the dataset BHCF_2020.
17. Download the 2021 bank holding company reports and import the dataset into R using the following link <https://www.dropbox.com/s/1ag5159cle920yg/BHCF20211231.csv?dl=1>. Name the dataset BHCF_2021.
18. Using the data dictionary in Appendix A, select the following variables for each data frame, i.e., each year (please note that the code for some variables is not consistent across all years):
 - a. RSSD Number
 - b. Name
 - c. Date
 - d. Loans held for investment
 - e. Loans held for sale
 - f. Allowance for loan losses
 - g. Non-performing loans
 - h. Total assets
 - i. Non-interest-bearing deposits

- j. Money market and savings deposits
 - k. Total equity
 - l. Net interest income
 - m. Fiduciary income
 - n. Security gains (held-to-maturity (HTM))
 - o. Security gains (available-for-sale (AFS))
 - p. Trading gains
 - q. Non-interest expense
 - r. Salary and employee benefits expense
 - s. Income before taxes
 - t. Taxes
 - u. Net income
 - v. Number of employees
19. If variable names are inconsistent across all years, the variable will need to be renamed to combine the data frames using the rbind function. An error message will be encountered if the rbind function is used and the variable names are inconsistent.
 - a. Rename variables if necessary.
 20. Combine all BHCF Data Frames using the rbind function and create a new data frame named BHCF.
 21. Convert the date column to three columns: Year, Month, and Day.
 22. Convert all columns except RSSD9010 to numeric.
 23. Drop observations with a “na” in value.
 24. Filter out BHCs that do not have assets.
 25. Calculate the variables listed in Table 1 (Appendix)
 26. If the calculations in requirement 25 created “na’s,” remove those observations.
 27. Load the Fredr library.
 28. You will need an account and API key to import FRED data. The account and API key can be obtained from https://fred.stlouisfed.org/docs/api/api_key.html. Accounts and API keys are free for educational and research purposes.
 - a. Enter your API Key
 29. Import United States real GDP beginning in January 2011 data from FRED.
 30. Select date and GDP columns
 31. Rename the value column to GDP
 32. Separate the date column into three columns using the splitstackshape library and change headings to year, month, and day
 33. Calculate the annual average GDP.
 34. Calculate the annual growth rate
 35. Remove observations with “na’s.”
 36. Import non-farm payrolls beginning with January 2011.
 37. Select date and value columns
 38. Change value column name to NFP
 39. Using the splitstackshape library, separate the date column into three columns and change headings to year, month, and day
 40. Calculate the annual average non-farm payroll
 41. Calculate the annual non-farm payroll growth rate
 42. Remove “na’s.”

43. Import the CPI data from FRED beginning in January 2011.
44. Select the date and value columns.
45. Change the name of the value column to CPI
46. Separate the date column into three columns and change headings to year, month and day using the splitstackshape package.
47. Calculate the annual average CPI.
48. Calculate the annual CPI growth rate.
49. Remove “na’s.”
50. Import the treasury spread (the ten-year rate minus the 3-month rate) beginning in January 2011.
51. Select the date and value columns.
52. Change value column name to Spread.
53. Separate the date column into three columns and change headings to year, month and day using the splitstackshape package.
54. Drop na’s.
55. Calculate the annual average T_Spread.
56. Import the ten-year treasury rate beginning in 2011.
57. Select the value and date columns
58. Change the value column name to ten_yr_rate.
59. Separate the date column into three columns and change headings to year, month and day using the splitstackshape package.
60. Drop “na’s.”
61. Calculate the annual average ten-year rate.
62. Import Reg_Data using the following link
https://www.dropbox.com/s/4xya5vhm1m4yhbm/regdata_4_0_documents.csv?dl=1.
63. Separate year month, and day into separate variables.
64. Delete the first column.
65. Separate the document reference column into title and part; change column names.
66. Select title 12, i.e., delete everything that is not title 12. Title 12 applies to banks and banking.
67. Calculate totals by year for words, restrictions 2_0, restrictions 1_0, and conditionals; calculate averages by year for Flesch reading ease, sentence length, and Shannon’s entropy.
68. Calculate growth rate for words, restrictions 2.0, restrictions 1.0, conditionals, flesh reading ease, sentence length, and Shannon entropy
69. Convert year column/variable to numeric for NFP, CPI, RGDP, BHCF, T_spread, ten_yr, and reg_data_year.
70. Join the BHCF, NFP, RGDP, CPI, T_spread, ten_yr, reg_data_year data frames. Use the BHCF data frame name.

CASE TEACHING NOTES

2.1 Overview and learning outcomes

This case was written for and used in a graduate-level audit analytics course. Nevertheless, the stated learning objectives are appropriate for most classes seeking to incorporate analytics into their curriculum. We designed and wrote this case because most data sets used in academic courses are relatively small, clean, and focus on analysis and decision-making rather than the messy front-end process. Our personal experience has been that real datasets are rather messy and are not small. In addition, we have found that analysis is much less time-consuming and is relatively easy once data has been through the ETL processes. Our perceptions are supported by Raschke & Charron (2021). After completing the required published requirements, the instructor may ask students to perform statistical analysis. The case is very flexible. In addition, FRED has an immense number of datasets available for download. The instructor may ask students to select and download datasets other than or in addition to those listed in the case study.

The case is flexible, and all or part of the case may be assigned. Steps 1 – 26 are related to bank holding company datasets and can stand independently. Steps 27 – 61 relate to Federal Reserve economic datasets and can stand independently. Steps 62 – 69 relate to the RegData dataset and can stand independently. The last step in the case asks students to join all the datasets.

After joining all datasets, it is possible to ask students to prepare statistical analyses and data visualizations. In our graduate class, we asked students to complete basic statistical analysis and prepare simple visualizations. However, by doing this, the case extended well beyond one class period. We leave these additional steps to the discretion of each instructor.

2.1.1 Learning Objectives.

After completing this case, students will be able to use R to:

- Import large datasets.
- Identify specific variables using a data dictionary.
- Select specific variables.
- Combine multiple datasets.
- Create calculated variables.
- Filter out observations based on specified criteria.

While this case is focused on ETL, you may choose to ask students to visualize and/or analyze the data and demonstrate an ability to reach critical conclusions.

2.2 Implementation guidance

We utilized this case study in a graduate audit analytics class during the fall of 2021, but it is appropriate for any course seeking to include analytics in the curriculum. The class was taught in a computer lab, and students were encouraged to work together in small groups. An instructor was available throughout the course to answer questions and provide guidance as students encountered issues. In addition, students had access to the internet and were encouraged to search for answers. Students in this class had previously taken an accounting statistics class

that used R; however, they did not consider themselves R competent but were familiar with the syntax structure. The case could be assigned as homework if students are familiar with R.

2.3 Efficacy

As stated in the previous section, this case study was assigned to a class in a computer lab; students were encouraged to work together, search the internet, and ask the instructor questions. As a result, all students satisfactorily completed the assignment, and it was graded “complete.” Some students were very efficient and completed the project without significant help. Other students required considerable guidance.

2.4 Student perceptions

One student said, “I wish I knew how to do this when I was in my internship. It would have saved me so much time.” Another student complained that they were not allowed to complete the assignment in Excel. This student felt that her familiarity with Excel would have made completing the case much quicker and easier.

2.5 Requirements and Proposed Solution

We have provided the requirements, followed by a solution and example code. The code provided has been written and tested on both windows and mac operating systems and ran without error when the case was written; however, it should be noted that as packages are updated, it may be necessary to adjust the code. We use the term “a solution” because R is highly flexible, and there are often multiple ways to accomplish a task. The code we provide in the solution is not the only solution, and students may use other correct methods. Code is in bold and should be typed as it is presented. The entire code may be downloaded as an R-Script file using the following links https://www.dropbox.com/s/lgl1ujtbctqgthv/BHCF_Case.R?dl=0 or as a *.txt file using this link https://www.dropbox.com/s/bt3aqtuk5t1v68o/bhc_code.txt?dl=0. The code at these links assumes R, RStudio, and the required packages are already installed on your computer. This code does not include a FRED API key.

1. If R and/or RStudio are not installed on your computer, download, and install R and RStudio.
 - a. Download R from <https://www.r-project.org/>.
 - i. Follow onscreen instructions.
 - b. Download RStudio from <https://www.rstudio.com/products/rstudio/>.
 - i. Follow onscreen instructions.
2. Install the Tidyverse Package
`install.packages("tidyverse")3`
3. Install the Lubridate Package
`install.packages("lubridate")3`

³ Since it is only necessary to install packages once, it may be preferable to type this code in the console section of RStudio and not save it a part of your script. Otherwise, you may choose to put “#” in front of it after installing it to prevent it from running each time you run your script. It is also possible to type it in the script file and then delete it after it runs.

4. Install the Fredr Package
`install.packages("fredr")`**Error! Bookmark not defined.**
5. Install the Splitstackshape Package
`install.packages("splitstackshape")`
6. Load the Tidyverse, Lubridate, Fredr, and Splitstackshape packages.
`library(tidyverse)`
`library(lubridate)`
`library(fredr)`
7. Download the 2011 bank holding company financial reports and import the dataset into R using the following link <https://www.chicagofed.org/~media/others/banking/financial-institution-reports/bhc-data/bhcf1112.csv>. Create a data frame named BHCF_2011.
`BHCF_2011 <-`
`read.csv("https://www.chicagofed.org/~media/others/banking/financial-institution-reports/bhc-data/bhcf1112.csv")`
8. Download the 2012 bank holding company financial reports and import the dataset into R using the following link <https://www.chicagofed.org/~media/others/banking/financial-institution-reports/bhc-data/bhcf1212.csv>. Create a data frame named BHCF_2012.
 - a. Modify the code in step 7 to download 2012, i.e., change BHCF_2011 to BHCF_2012 and replace the HTTPS link in step 7 with the HTTPS link provided in step 8.
9. Download the 2013 bank holding company financial reports and import the dataset into R using the following link <https://www.chicagofed.org/~media/others/banking/financial-institution-reports/bhc-data/bhcf1312.csv>. Create a data frame named BHCF_2013.
 - a. Modify the code in step 7 to download 2013, i.e., change BHCF_2011 to BHCF_2013 and replace the HTTPS link in step 7 with the HTTPS link provided in step 8.
10. Download the 2014 bank holding company financial reports and import the dataset into R using the following link <https://www.chicagofed.org/~media/others/banking/financial-institution-reports/bhc-data/bhcf1412.csv>. Create a data frame named BHCF_2014.
 - a. Modify the code in step 7 to download 2014, i.e., change BHCF_2011 to BHCF_2014 and replace the HTTPS link in step 7 with the HTTPS link provided in step 8.
11. Download the 2015 bank holding company financial reports and import the dataset into R using the following link <https://www.chicagofed.org/~media/others/banking/financial-institution-reports/bhc-data/bhcf1512.csv>. Create a data frame named BHCF_2015.
 - a. Modify the code in step 7 to download 2015, i.e., change BHCF_2011 to BHCF_2015 and replace the HTTPS link in step 7 with the HTTPS link provided in step 8.
12. Download the 2016 bank holding company reports and import the dataset into R using the following link <https://www.chicagofed.org/~media/others/banking/financial-institution-reports/bhc-data/bhcf1612.csv>. Name the dataset BHCF_2016.
 - a. Modify the code in step 7 to download 2016, i.e., change BHCF_2011 to BHCF_2016 and replace the HTTPS link in step 7 with the HTTPS link provided in step 8.

13. Download the 2017 bank holding company reports and import the dataset into R using the following link <https://www.chicagofed.org/~media/others/banking/financial-institution-reports/bhc-data/bhcf1712.csv>. Name the dataset BHCF_2017.
 - a. Modify the code in step 7 to download 2017, i.e., change BHCF_2011 to BHCF_2012 and replace the HTTPS link in step 7 with the HTTPS link provided in step 8.
14. Download the 2018 bank holding company reports and import the dataset into R using the following link <https://www.chicagofed.org/~media/others/banking/financial-institution-reports/bhc-data/bhcf1812.csv>. Name the dataset BHCF_2018.
 - a. Modify the code in step 7 to download 2018, i.e., change BHCF_2011 to BHCF_2012 and replace the HTTPS link in step 7 with the HTTPS link provided in step 8.
15. Download the 2019 bank holding company reports and import the dataset into R using the following link <https://www.chicagofed.org/~media/others/banking/financial-institution-reports/bhc-data/bhcf1912.csv>. Name the dataset BHCF_2019.
 - a. Modify the code in step 7 to download 2019, i.e., change BHCF_2011 to BHCF_2019 and replace the HTTPS link in step 7 with the HTTPS link provided in step 8.
16. Download the 2020 bank holding company reports and import the dataset into R using the following link <https://www.chicagofed.org/~media/others/banking/financial-institution-reports/bhc-data/bhcf2012.csv>. Name the dataset BHCF_2020.
 - a. Modify the code in step 7 to download 2020, i.e., change BHCF_2011 to BHCF_2020 and replace the HTTPS link in step 7 with the HTTPS link provided in step 8.
17. Download the 2021 bank holding company reports and import the dataset into R using the following link <https://www.dropbox.com/s/1ag5159cle920yg/BHCF20211231.csv?dl=1>. Name the dataset BHCF_2021.
 - a. Modify the code in step 7 to download 2021, i.e., change BHCF_2011 to BHCF_2021 and replace the HTTPS link in step 7 with the HTTPS link provided in step 8.
 - b. After completing steps 7 – 17, students should have a data frame for each year, see Figure 1. A quick review reveals that the data frames are different shapes and numbers of observations and variables. The number of bank holding companies varies per year, and the data collected varies by year.

[Insert Figure 1]
 - c. We intend to combine the 11 data frames using the append function. If you visualize the data frames as spreadsheets, the append function will add each successive year to the bottom of the previous data frame. To do this, they must have the same number of variables (columns), and the variables must have the same names. Therefore, students must select the desired variables and verify that the variable names are consistent for each year.
18. Using the data dictionary in Appendix A, select the following variables for each data frame (please note that the code for some variables is not consistent across all years):
 - a. RSSD Number
 - b. Name

- c. Date
- d. Loans held for investment
- e. Loans held for sale
- f. Allowance for loan losses
- g. Non-performing loans
- h. Total assets
- i. Non-interest-bearing deposits
- j. Money market and savings deposits
- k. Total equity
- l. Net interest income
- m. Fiduciary income
- n. Security gains (held-to-maturity (HTM))
- o. Security gains (available-for-sale (AFS))
- p. Trading gains
- q. Non-interest expense
- r. Salary and employee benefits expense
- s. Income before taxes
- t. Taxes
- u. Net income
- v. Number of employees

This will need to be completed for each year from 2011 to 2021. The code for 2011 is provided in Figure 2.

[Insert Figure 2]

19. Since the variable name for non-performing loans differs from 2011 to 2017 and 2018 to 2021, it is necessary to rename the variable to be consistent across all data frames. Otherwise, students will receive an error message when combining the data frames.
 - a. For 2011-2017 rename BHCK5526 to NONPERF
 - i. `BHCF_2011 <- BHCF_2011 %>%
rename(NONPERF = BHCK5526)`
 - ii. Repeat for each data frame 2012 - 2017
 - b. For 2018-2021 rename BHCK1403 to NONPERF
 - i. `BHCF_2018 <- BHCF_2018 %>%
rename(NONPERF = BHCK1403)`
 - ii. Repeat for each data frame 2018 – 2021.
20. Combine all BHCF Data Frames using the rbind function and create a new data frame called BHCF.
 - a. `BHCF <- rbind(BHCF_2011,
BHCF_2012,
BHCF_2013,
BHCF_2014,
BHCF_2015,
BHCF_2016,
BHCF_2017,
BHCF_2018,
BHCF_2019,
BHCF_2020,`

BHCF_2021)

21. Convert the date column to three columns: Year, Month, and Day.

a. `BHCF <- extract(BHCF, RSSD9999, into = c("Year", "Month", "Day"),
"(.{4}).{2}).{2}")`

22. Note that all variables, including numeric columns, are recognized as characters; see Figure 3. To perform calculations, it is necessary to convert the numeric columns to numeric. Convert all columns except RSSD9010 to numeric.

a. `BHCF <- BHCF %>%
mutate_at(c('RSSD9001',
'Year',
'Month',
'Day',
'BHCKB528',
'BHCK5369',
'BHCK3123',
'NONPERF',
'BHCK2170',
'BHCB2210',
'BHCB3187',
'BHCB2389',
'BHCKG105',
'BHCK4074',
'BHCK4070',
'BHCK3521',
'BHCK3196',
'BHCKA220',
'BHCK4093',
'BHCK4135',
'BHCK4301',
'BHCK4302',
'BHCK4340',
'BHCK4150'), as.numeric)`

After conversion, all columns except RSSD9010 are numeric; see Figure 4.

[Insert Figure 3]

[Insert Figure 4]

23. Replace na's with 0's.

a. `BHCF <- drop_na(BHCF)`

24. Filter out BHC's that do not have assets.

a. `BHCF <- filter(BHCF, BHCK2170 >.01)`

25. Calculate the variables listed in Table 1, see below.

```
BHCF <- BHCF %>%  
  arrange(RSSD9001, Year) %>%  
  group_by(RSSD9010) %>%  
  mutate(AVG_ASST = (BHCK2170 + lag(BHCK2170, order_by = Year))/2)  
  %>%
```

- ```

mutate(AVG_EQ = (BHCKG105 + lag(BHCKG105, order_by = Year))/2) %>%
mutate(PRE_ROA = BHCK4301/AVG_ASST) %>%
mutate(PRE_ROE = BHCK4301/AVG_EQ) %>%
mutate(TOT_LNS = BHCKB528 + BHCK5369) %>%
mutate(LNS_PCT = ((TOT_LNS)/BHCK2170) * 100) %>%
mutate(LNS_EMP = TOT_LNS/BHCK4150) %>%
mutate(TRANS_DEP_PCT = ((BHCB2210 + BHCB3187 +
BHCB2389)/BHCK2170) * 100) %>%
mutate(SAL_PCT = (BHCK4135/BHCK2170) * 100) %>%
mutate(AVG_Pay = (BHCK4135/BHCK4150) * 100) %>%
mutate(SEC_GAINS = BHCK3521 + BHCK3196 + BHCKA220) %>%
mutate(CAP_RATIO = (BHCKG105/BHCK2170) * 100) %>%
mutate(OTHER = BHCK4093 - BHCK4135)

```
26. If the calculations in requirement 25 created na's, remove those observations.
    - a. `BHCF <- drop_na(BHCF)`
  27. Load the Fredr library.
    - a. `library(fredr)`
  28. You will need an account and API key to import FRED data, which can be obtained from [https://fred.stlouisfed.org/docs/api/api\\_key.html](https://fred.stlouisfed.org/docs/api/api_key.html). Accounts and API keys are free for educational and research purposes.
    - a. Set your API Key
      - i. `fredr_set_key("*****")`
  29. Import United States real GDP beginning in 2011 data from FRED.
    - a. `RGDP <- fredr(series_id = "GDPC1", observation_start = as.Date("2011-01-01"))`
  30. Select date and GDP columns
    - a. `RGDP <- select(RGDP, c(date, value))`
  31. Rename the value column to GDP
    - a. `colnames(RGDP)[colnames(RGDP) == "value"] <- "GDP"`
  32. Separate the date column into three columns using the splitstackshape library and change headings to Year, month and day
    - a. `RGDP <- splitstackshape::cSplit(indt = RGDP, splitCols = 'date', sep = '-', type.convert = F)`
      - b. `colnames(RGDP)[colnames(RGDP) == "date_1"] <- "Year"`
      - c. `colnames(RGDP)[colnames(RGDP) == "date_2"] <- "month"`
      - d. `colnames(RGDP)[colnames(RGDP) == "date_3"] <- "day"`
  33. Calculate the annual average GDP.
    - a. `RGDP <- RGDP %>%
group_by(Year) %>%
summarise_at(vars(GDP), list(ANN_GDP = mean))`
  34. Calculate annual growth rate
    - a. `RGDP <- RGDP %>%
mutate(GDP_grow = (((Annual_GDP -
lag(Annual_GDP))/lag(Annual_GDP))) * 100)`

35. Remove observations with na's.
  - a. `RGDP <- drop_na(RGDP)`
36. Import non-farm payrolls beginning with 2011.
  - a. `NFP <- fredr(series_id = "PAYEMS", observation_start = as.Date("2011-01-01"))`
37. Select date and value columns
  - a. `NFP <- select(NFP, c(date, value))`
38. Change value column name to NFP
  - a. `colnames(NFP)[colnames(NFP) == "value"] <- "NFP"`
39. Using the splitstackshape library, separate the date column into three columns and change headings to Year, month, and day
  - a. `NFP <- splitstackshape::cSplit(indt = NFP, splitCols = 'date', sep = '-', type.convert = F)`
  - b. `colnames(NFP)[colnames(NFP) == "date_1"] <- "Year"`
  - c. `colnames(NFP)[colnames(NFP) == "date_2"] <- "month"`
  - d. `colnames(NFP)[colnames(NFP) == "date_3"] <- "day"`
40. Calculate the annual average non-farm payroll
  - a. `NFP <- NFP %>%  
group_by(Year) %>%  
summarise_at(vars(NFP), list(Annual_NFP = mean))`
41. Calculate the annual non-farm payroll growth rate
  - a. `NFP <- NFP %>%  
mutate(NFP_grow = (((Annual_NFP -  
lag(Annual_NFP))/lag(Annual_NFP))) * 100)`
42. Remove na's
  - a. `NFP <- drop_na(NFP)`
43. Import the CPI data from FRED beginning in 2011.
  - a. `CPI <- fredr(series_id = "CPALTT01USA661S", observation_start = as.Date("2011-01-01"))`
44. Select date and value columns.
  - a. `CPI <- select(CPI, c(date, value))`
45. Change the name of the value column to CPI
  - a. `colnames(CPI)[colnames(CPI) == "value"] <- "CPI"`
46. Separate the date column into three columns and change headings to Year, month and day using the splitstackshape package.
  - a. `CPI <- splitstackshape::cSplit(indt = CPI, splitCols = 'date', sep = '-', type.convert = F)`
  - b. `colnames(CPI)[colnames(CPI) == "date_1"] <- "Year"`
  - c. `colnames(CPI)[colnames(CPI) == "date_2"] <- "month"`
  - d. `colnames(CPI)[colnames(CPI) == "date_3"] <- "day"`
47. Calculate the annual average CPI.
  - a. `CPI <- CPI %>%  
group_by(Year) %>%`

- ```

summarise_at(vars(CPI), list(Annual_CPI = mean))

```
48. Calculate the annual CPI growth rate.
 - a. `CPI <- CPI %>% mutate(CPI_grow = (((Annual_CPI - lag(Annual_CPI))/lag(Annual_CPI))) * 100)`
 49. Remove na's
 - a. `CPI <- drop_na(CPI)`
 50. Import the treasury spread (ten-year – 3-month rate) beginning in 2011.
 - a. `T_spread <- fredr(series_id = "T10Y3M", observation_start = as.Date("2011-01-01"))`
 51. Select the date and value columns.
 - a. `T_spread <- select(T_spread, c(date, value))`
 52. Change value column name to Spread.
 - a. `colnames(T_spread)[colnames(T_spread) == "value"] <- "Spread"`
 53. Separate the date column into three columns and change headings to Year, month and day using the splitstackshape package.
 - a. `T_spread <- splitstackshape::cSplit(indt = T_spread, splitCols = 'date', sep = '-', type.convert = F)`
 - b. `colnames(T_spread)[colnames(T_spread) == "date_1"] <- "Year"`
 - c. `colnames(T_spread)[colnames(T_spread) == "date_2"] <- "month"`
 - d. `colnames(T_spread)[colnames(T_spread) == "date_3"] <- "day"`
 54. Drop na's.
 - a. `T_spread <- drop_na(T_spread)`
 55. Calculate the annual average T_spread.
 - a. `T_spread <- T_spread %>%
 group_by(Year) %>%
 summarise_at(vars(Int_Slope), list(Annual_Slope = mean))`
 56. Import the ten year treasury rate beginning in 2011.
 - a. `ten_yr <- fredr(series_id = "DGS10", observation_start = as.Date("2011-01-01"))`
 57. Select the value and date columns
 - a. `ten_yr <- select(ten_yr, c(date, value))`
 58. Change the value column name to ten_yr_rate.
 - a. `colnames(ten_yr)[colnames(ten_yr) == "value"] <- "ten_yr_rate"`
 59. Separate the date column into three columns and change headings to Year, month and day using the splitstackshape package.
 - a. `ten_yr <- splitstackshape::cSplit(indt = ten_yr, splitCols = 'date', sep = '-', type.convert = F)`
 - b. `colnames(ten_yr)[colnames(ten_yr) == "date_1"] <- "Year"`
 - c. `colnames(ten_yr)[colnames(ten_yr) == "date_2"] <- "month"`
 - d. `colnames(ten_yr)[colnames(ten_yr) == "date_3"] <- "day"`
 60. Drop na's.
 - a. `ten_yr <- drop_na(ten_yr)`

61. Calculate annual average ten year rate.
- `ten_yr <- ten_yr %>%
group_by(Year) %>%
summarise_at(vars(ten_yr_rate), list(Annual_10yr = mean))`
62. Import Reg_Data
- `reg_data <-
read.csv("https://www.dropbox.com/s/4xya5vhm1m4yhbm/regdata_4_0_documents.csv?dl=1")`
63. Separate year month and day into separate variables
- `reg_data <- splitstackshape::cSplit(indt = reg_data, splitCols = 'Year', sep = '-',
type.convert = FALSE)`
 - `colnames(reg_data)[colnames(reg_data) == "Year_1"] <- "Year"`
 - `colnames(reg_data)[colnames(reg_data) == "Year_2"] <- "month"`
 - `colnames(reg_data)[colnames(reg_data) == "Year_3"] <- "day"`
64. Delete the first column
- `reg_data$X <- NULL`
65. Separate the document reference column into title and part; change column names.
- `reg_data <- splitstackshape::cSplit(indt = reg_data, splitCols =
'document_reference', sep = ',', type.convert = FALSE)`
 - `colnames(reg_data)[colnames(reg_data) == "document_reference_1"] <- "title"`
 - `colnames(reg_data)[colnames(reg_data) == "document_reference_2"] <- "part"`
 - `reg_data <- splitstackshape::cSplit(indt = reg_data, splitCols = 'title', sep = ',',
type.convert = FALSE)`
 - `reg_data$title_1 <- NULL`
 - `colnames(reg_data)[colnames(reg_data) == "title_2"] <- "title"`
 - `reg_data <- splitstackshape::cSplit(indt = reg_data, splitCols = 'part', sep = ',',
type.convert = FALSE)`
 - `reg_data$part_1 <- NULL`
 - `colnames(reg_data)[colnames(reg_data) == "part_2"] <- "part"`
66. Select title 12, i.e., delete everything that is not title 12. Title 12 applies to banks and banking.
- `reg_data <- filter(reg_data, title == "12")`
67. Calculate totals by year for words, restrictions 2_0, restrictions 1_0, and conditionals; calculate averages by year for flesch reading ease, sentence length and shannons entropy.
- `reg_data_year <- reg_data %>%
group_by(Year) %>%
summarise(words = sum(words),
restrictions_2_0 = sum(restrictions_2_0),
restrictions_1_0 = sum(restrictions_1_0),
conditionals = sum(conditionals),`

```
flesch_reading_ease = mean(flesch_reading_ease),
sentence_length = mean(sentence_length),
shannon_entropy = mean(shannon_entropy))
```

68. Calculate growth rate for words, restrictions 2.0, restrictions 1.0, conditionals, flesh reading ease, sentence length, and shannon entropy
- ```
reg_data_year <- reg_data_year %>%
 mutate(words_grow = (((words - lag(words))/lag(words))) * 100) %>%
 mutate(restrictions_2_0_grow = (((restrictions_2_0 -
 lag(restrictions_2_0))/lag(restrictions_2_0))) * 100) %>%
 mutate(restrictions_1_0_grow = (((restrictions_1_0 -
 lag(restrictions_1_0))/lag(restrictions_1_0))) * 100) %>%
 mutate(conditionals_grow = (((conditionals -
 lag(conditionals))/lag(conditionals))) * 100) %>%
 mutate(flesch_reading_ease_grow = (((flesch_reading_ease -
 lag(flesch_reading_ease))/lag(flesch_reading_ease))) * 100) %>%
 mutate(sentence_length_grow = (((sentence_length -
 lag(sentence_length))/lag(sentence_length))) * 100) %>%
 mutate(shannon_entropy_grow = (((shannon_entropy -
 lag(shannon_entropy))/lag(shannon_entropy))) * 100)
```
69. Convert year column/variable to numeric for NFP, CPI, RGDP, BHCF, T\_spread, ten\_yr, and reg\_data\_year.
- NFP\$Year <- as.numeric(NFP\$Year)
  - CPI\$Year <- as.numeric(CPI\$Year)
  - RGDP\$Year <- as.numeric(RGDP\$Year)
  - reg\_data\_Year\$Year <- as.numeric(reg\_data\_Year\$Year)
  - BHCF\$Year <- as.numeric(BHCF\$Year)
  - T\_spread\$Year <- as.numeric(T\_spread\$Year)
  - ten\_yr\$Year <- as.numeric(ten\_yr\$Year)
70. Join the BHCF, NFP, RGDP, CPI, T\_spread, ten\_yr, reg\_data\_year data frames. Use the BHCF data frame name.
- BHCF <- inner\_join(BHCF, NFP, by = "Year")
  - BHCF <- inner\_join(BHCF, RGDP, by = "Year")
  - BHCF <- inner\_join(BHCF, CPI, by = "Year")
  - BHCF <- inner\_join(BHCF, T\_spread, by = "Year")
  - BHCF <- inner\_join(BHCF, ten\_yr, by = "Year")
  - BHCF <- inner\_join(BHCF, reg\_data\_year, by = "Year")

## Figures

| Environment               | History | Connections                 | Tutorial |
|---------------------------|---------|-----------------------------|----------|
| Import Dataset   1.42 GiB |         |                             |          |
| Global Environment        |         |                             |          |
| Data                      |         |                             |          |
| BHCF_2011                 |         | 5148 obs. of 2268 variables |          |
| BHCF_2012                 |         | 5481 obs. of 2284 variables |          |
| BHCF_2013                 |         | 5321 obs. of 2327 variables |          |
| BHCF_2014                 |         | 5111 obs. of 2382 variables |          |
| BHCF_2015                 |         | 4895 obs. of 2482 variables |          |
| BHCF_2016                 |         | 4751 obs. of 2459 variables |          |
| BHCF_2017                 |         | 4599 obs. of 2486 variables |          |
| BHCF_2018                 |         | 4410 obs. of 2259 variables |          |
| BHCF_2019                 |         | 4237 obs. of 2328 variables |          |
| BHCF_2020                 |         | 4118 obs. of 2364 variables |          |
| BHCF_2021                 |         | 3977 obs. of 2321 variables |          |

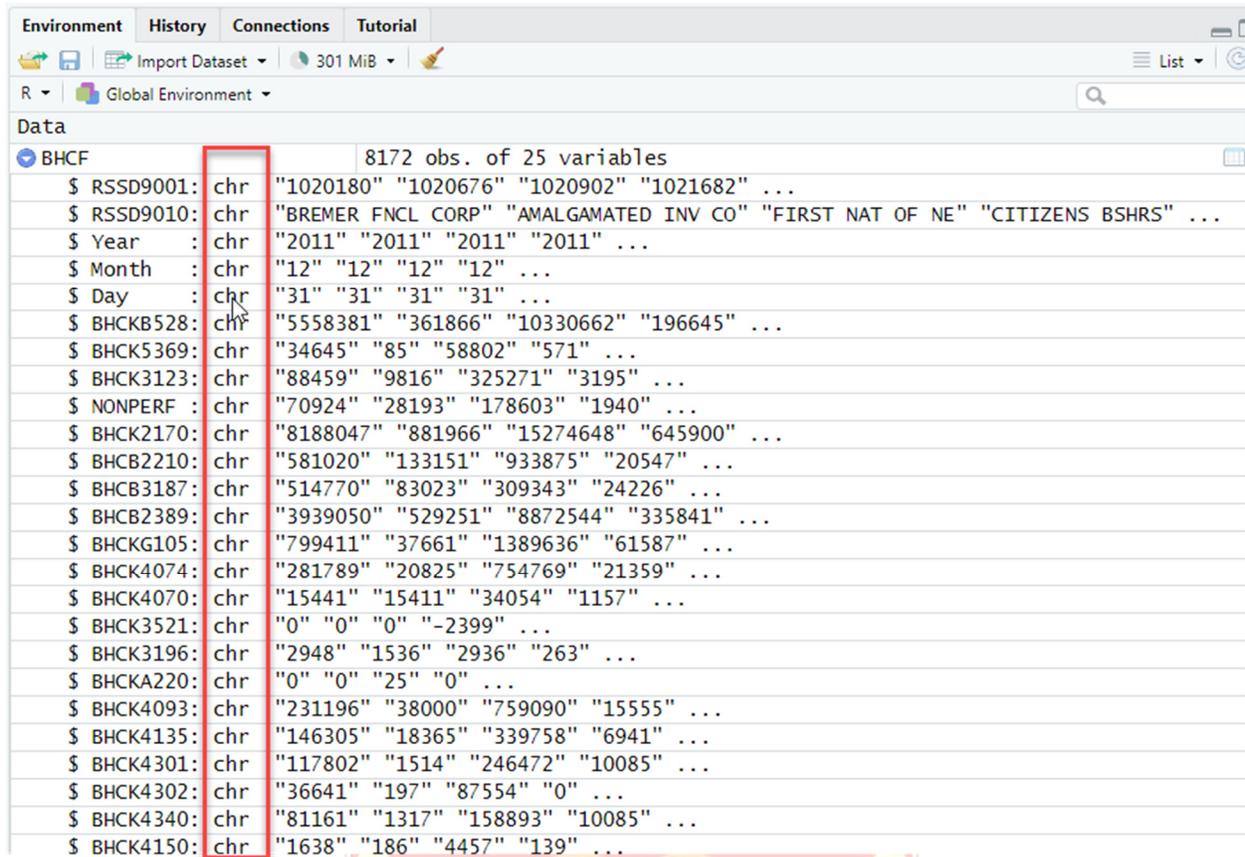
Figure 1: Data Frames Created from Requirements 7 – 17

```

37
38 #Select specified variables -- BHCF_2011
39 BHCF_2011 <- select(BHCF_2011, c(RSSD9001,
40 RSSD9010,
41 RSSD9999,
42 BHCKB528,
43 BHCK5369,
44 BHCK3123,
45 BHCK5526,
46 BHCK2170,
47 BHCB2210,
48 BHCB3187,
49 BHCB2389,
50 BHCKG105,
51 BHCK4074,
52 BHCK4070,
53 BHCK3521,
54 BHCK3196,
55 BHCKA220,
56 BHCK4093,
57 BHCK4135,
58 BHCK4301,
59 BHCK4302,
60 BHCK4340,
61 BHCK4150))
62
63

```

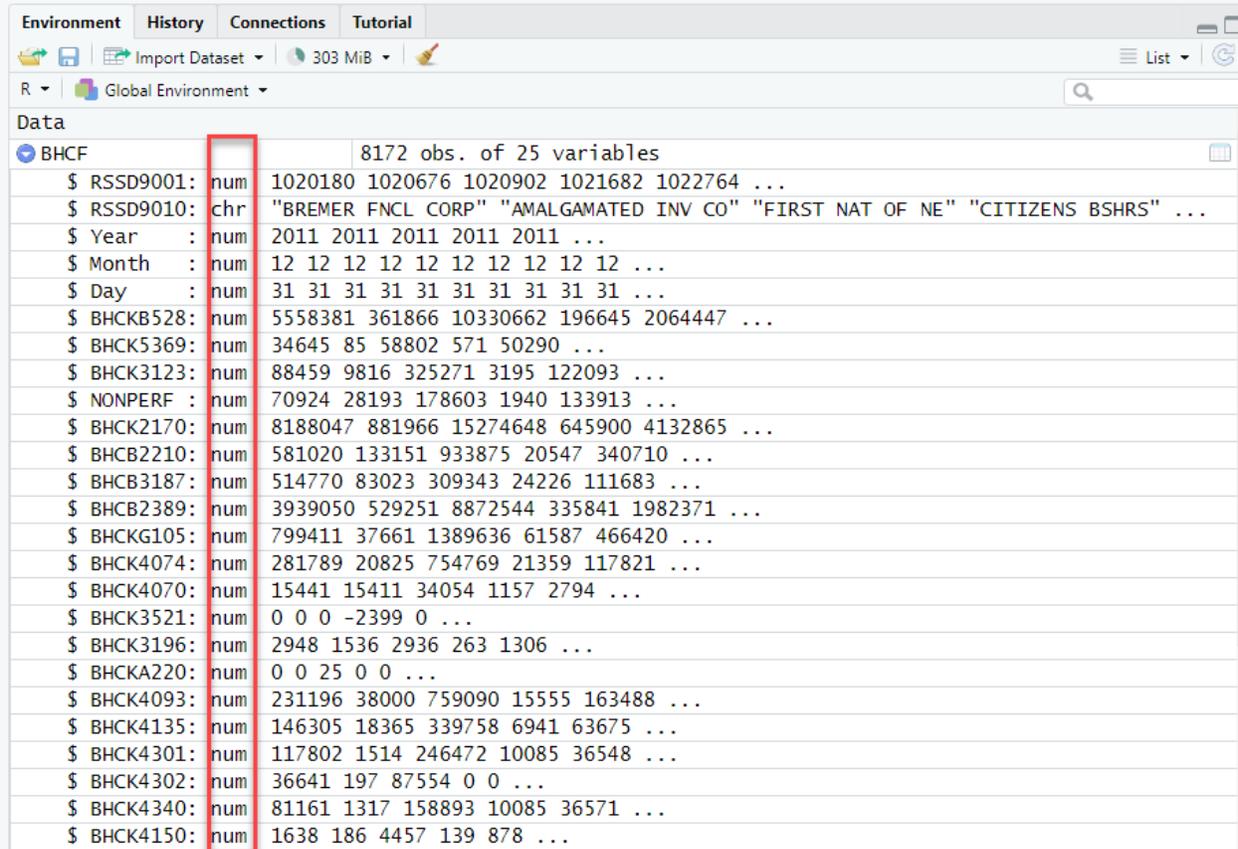
Figure 2: Select Variables



The screenshot shows the RStudio interface with a data table named 'BHCF' containing 8172 observations and 25 variables. The first column of the table is highlighted with a red box, indicating the selection of variables. The variables listed in the first column are:

| Variable Name | Variable Type |
|---------------|---------------|
| \$ RSSD9001   | chr           |
| \$ RSSD9010   | chr           |
| \$ Year       | chr           |
| \$ Month      | chr           |
| \$ Day        | chr           |
| \$ BHCKB528   | chr           |
| \$ BHCK5369   | chr           |
| \$ BHCK3123   | chr           |
| \$ NONPERF    | chr           |
| \$ BHCK2170   | chr           |
| \$ BHCB2210   | chr           |
| \$ BHCB3187   | chr           |
| \$ BHCB2389   | chr           |
| \$ BHCKG105   | chr           |
| \$ BHCK4074   | chr           |
| \$ BHCK4070   | chr           |
| \$ BHCK3521   | chr           |
| \$ BHCK3196   | chr           |
| \$ BHCKA220   | chr           |
| \$ BHCK4093   | chr           |
| \$ BHCK4135   | chr           |
| \$ BHCK4301   | chr           |
| \$ BHCK4302   | chr           |
| \$ BHCK4340   | chr           |
| \$ BHCK4150   | chr           |

Figure 3: Data Type



The screenshot shows the RStudio interface with a data table. The table has 25 columns and 8172 rows. The first column is labeled 'BHC' and contains variable names. The second column is labeled 'num' and contains the data type for each variable. The remaining columns contain sample data values. A red box highlights the 'num' column and the first few rows of data.

| Variable | Type | Sample Values                                                                  |
|----------|------|--------------------------------------------------------------------------------|
| RSSD9001 | num  | 1020180 1020676 1020902 1021682 1022764 ...                                    |
| RSSD9010 | chr  | "BREMER FNCL CORP" "AMALGAMATED INV CO" "FIRST NAT OF NE" "CITIZENS BSHRS" ... |
| Year     | num  | 2011 2011 2011 2011 2011 ...                                                   |
| Month    | num  | 12 12 12 12 12 12 12 12 12 ...                                                 |
| Day      | num  | 31 31 31 31 31 31 31 31 31 ...                                                 |
| BHCKB528 | num  | 5558381 361866 10330662 196645 2064447 ...                                     |
| BHCK5369 | num  | 34645 85 58802 571 50290 ...                                                   |
| BHCK3123 | num  | 88459 9816 325271 3195 122093 ...                                              |
| NONPERF  | num  | 70924 28193 178603 1940 133913 ...                                             |
| BHCK2170 | num  | 8188047 881966 15274648 645900 4132865 ...                                     |
| BHCB2210 | num  | 581020 133151 933875 20547 340710 ...                                          |
| BHCB3187 | num  | 514770 83023 309343 24226 111683 ...                                           |
| BHCB2389 | num  | 3939050 529251 8872544 335841 1982371 ...                                      |
| BHCKG105 | num  | 799411 37661 1389636 61587 466420 ...                                          |
| BHCK4074 | num  | 281789 20825 754769 21359 117821 ...                                           |
| BHCK4070 | num  | 15441 15411 34054 1157 2794 ...                                                |
| BHCK3521 | num  | 0 0 0 -2399 0 ...                                                              |
| BHCK3196 | num  | 2948 1536 2936 263 1306 ...                                                    |
| BHCKA220 | num  | 0 0 25 0 0 ...                                                                 |
| BHCK4093 | num  | 231196 38000 759090 15555 163488 ...                                           |
| BHCK4135 | num  | 146305 18365 339758 6941 63675 ...                                             |
| BHCK4301 | num  | 117802 1514 246472 10085 36548 ...                                             |
| BHCK4302 | num  | 36641 197 87554 0 0 ...                                                        |
| BHCK4340 | num  | 81161 1317 158893 10085 36571 ...                                              |
| BHCK4150 | num  | 1638 186 4457 139 878 ...                                                      |



Figure 4: Data Type After Conversion to Numeric

Table

Table 1: Calculated Variables with Basic Formula and New Variable Name

| Description                            | R Formula                                                           | New Variable Name |
|----------------------------------------|---------------------------------------------------------------------|-------------------|
| Average Assets                         | $BHCK2170 + \text{lag}(BHCK2170, \text{order\_by} = \text{Year})/2$ | AVG_ASST          |
| Average Equity                         | $BHCKG105 + \text{lag}(BHCKG105, \text{order\_by} = \text{Year})/2$ | AVG_EQ            |
| Pre-tax ROA                            | $BHCK4301/AVG\_ASST$                                                | PRE_ROA           |
| Pre-tax ROE                            | $BHCK4301/AVG\_EQ$                                                  | PRE_ROE           |
| Total loans                            | $(BHCKB528 + BHCK5369)$                                             | TOT_LNS           |
| Total loans as percent of total assets | $(TOT\_LNS / BHCK2170) * 100$                                       | LNS_PCT           |
| Loans per employee                     | $TOT\_LNS/BHCK4150$                                                 | LNS_EMP           |
| Total transaction deposits             | $((BHCB2210 + BHCB3187 + BHCB2389) / BHCK2170) * 100$               | TRANS_DEP_PCT     |
| Salaries to total assets               | $(BHCK4135/BHCK2170) * 100$                                         | SAL_PCT           |
| Average pay                            | $BHCK4135/BHCK4150$                                                 | AVG_PAY           |
| Total security gains                   | $BHCK3521+BHCK3196 +BHCKA220$                                       | SEC_GAINS         |
| Capital Ratio                          | $BHCKG105/BHCK2170$                                                 | CAP_RATIO         |
| Other non-interest expense             | $BHCK4093-BHCK4135$                                                 | OTHER             |

## Appendix A: Truncated Data Dictionary for BHCF

| Description                       | Applicable Years | Code     |
|-----------------------------------|------------------|----------|
| RSSD Number (Unique identifier)   | 2011-2021        | RSSD9001 |
| Name                              | 2011-2021        | RSSD9010 |
| Date                              | 2011-2021        | RSSD9999 |
| Loans held for investment         | 2011-2021        | BHCKB528 |
| Loans held for sale               | 2011-2021        | BHCK5369 |
| Allowance for loan losses         | 2011-2021        | BHCK3123 |
| Non-performing loans              | 2011-2017        | BHCK5526 |
| Non-performing loans              | 2018-2021        | BHCK1403 |
| Total Assets                      | 2011-2021        | BHCK2170 |
| Non-interest-bearing deposits     | 2011-2021        | BHCB2210 |
| Interest-bearing demand deposits  | 2011-2021        | BHCB3187 |
| Money market and savings deposits | 2011-2021        | BHCB2389 |
| Total Equity                      | 2011-2021        | BHCKG105 |
| Net interest income               | 2011-2021        | BHCK4074 |
| Fiduciary Income                  | 2011-2021        | BHCK4070 |
| Security gains (HTM)              | 2011-2021        | BHCK3521 |
| Security gains (AFS)              | 2011-2021        | BHCK3196 |
| Trading gains                     | 2011-2021        | BHCKA220 |
| Non-interest expense              | 2011-2021        | BHCK4093 |
| Salaries and employee benefits    | 2011-2021        | BHCK4135 |
| Income before taxes               | 2011-2021        | BHCK4301 |
| Taxes                             | 2011-2021        | BHCK4302 |
| Net Income                        | 2011-2021        | BHCK4340 |
| Number of Employees               | 2011-2021        | BHCK4150 |

**REFERENCES**

- American Institute of Certified Public Accountants (AICPA) 2020. Technology.* Retrieved June 13, 2022, from <https://www.aicpa.org/topic/technology>.
- fredr: An R Client for the “FRED” API version 2.1.0 from CRAN.* (n.d.). Retrieved April 28, 2022, from <https://rdr.io/cran/fredr/>
- Press, G. (2016, May 23). Cleaning Big Data: Most Time-Consuming, Least Enjoyable Data Science Task, Survey Says. *Forbes*.  
<https://www.forbes.com/sites/gilpress/2016/03/23/data-preparation-most-time-consuming-least-enjoyable-data-science-task-survey-says/>
- R Core Team. (2022). *R: A language and environment for statistical computing*. R Foundation for Statistical Computing. <https://www.R-project.org/>
- Raschke, R. L., & Charron, K. F. (2021). Review of Data Analytic Teaching Cases, Have We Covered Enough? *Journal of Emerging Technologies in Accounting*, 18(2), 247–255.  
<https://doi.org/10.2308/JETA-2020-036>

