California Military Economic Impact Study
Process Guide

Britnee Pannell & Sumeet Bedi

2022-01-24

Contents

Chapter 1

Introduction

The California Military Economic Impact Study report series began in 2018, when the California Governor's Office of Planning & Research (OPR) and the Governor's Military Council (GMC) requested the California Research Bureau (CRB, a unit of the California State Library) to conduct this study. This report provides detailed statewide and localized economic impacts of federal national security activity in the state of California. The federal agencies identified as related to national security includes the Departments of Defense (DoD), Homeland Security (DHS), Veterans Affairs (VA), and specified sub-agencies of the Department of Energy (DOE). The type of economic activity detailed includes spending (contracts, grants, veterans' benefits, and SmartPay charge card) and employment data (civilian and military). The first report, published in August 2018, utilized federal fiscal year 2016 data, while the second report, published in December 2019, used fiscal year 2018 data.

Following these 2 reports, OPR and CRB secured grant funding from DoD to support two full-time equivalent (FTE) positions to develop supplemental reports that help localize the economic impacts detailed in the statewide report. In December 2020, CRB produced the 2020 California Military Economic Impact Study, and followed this third edition of the statewide report with two first-time supplements that discuss the economic impacts in every county and congressional district in California. In December 2021, CRB completed the fourth version of the statewide report as well as the second edition of the county and congressional district supplements. After completion of the second round of supplements, CRB was tasked with producing this process guide document.

This process guide and supporting documentation were developed in order to allow other states to replicate the methodology of this study for their respective geography of interest. Additionally, this documentation serves to provide the rationale behind how the data was gathered, wrangled, and analyzed in order to justify the conclusions in our reports.

Chapter 2

Software Requirements

To recreate or replicate this study, specialized software is needed to obtain and process federal data. Fortunately, all the software that was used in this process is free and available online. It is best practice to use the most updated software version.

1. The R coding language from cloud.r-project.org. This language is used to obtain and process data.
2. RStudio Desktop from rstudio.com. RStudio is the integrated development environment (IDE) used to run R scripts and develop code.
3. Git from git-scm.com. A comprehensive guide to installing Git is available at happygitwithr.com. Git allows version control of edits across a multi-person team of researchers.
4. If a GitHub account is needed, one can sign up and register at https://github.com/. Individual free plans are available, as well as free upgrades for qualifying academic purposes. GitHub is used to develop and host this project.

2.1 Download Necessary R Libraries

To make sure the code can be run, please ensure that the following R library packages are installed. If you are using RStudio, packages can be installed from CRAN using the "install.packages(*[package_name]*)" command in the terminal.

- **dplyr:** Used for manipulating dataframes. (Wickham et al., 2021)
- **httr:** Used for working with URLs and HTTP. (Wickham, 2020)

- **jsonlite:** Used for interacting with JSON data and web APIs. (Ooms, 2014)
- **openxlsx:** Used for reading, writing and editing xlsx files. (Schauberger and Walker, 2021)
- **readxl:** Used for legacy excel .xls files. (Wickham and Bryan, 2019)
- **tidyverse:** A collection of R packages optimized for data science. (Wickham et al., 2019)

The next section details data necessary to complete this project as well as how to obtain it.

Chapter 3

Data Requirements

This section details how to obtain all the necessary data for this project, which we broke out into 3 types: 1) employment, 2) USAspending, and 3) all other additional data. *There is no guarantee that the data sources described below will exist in this form indefinitely, so care will be taken to keep this document as up to date as possible.*

3.1 Employment Data

This report focused on two types of employment data: civilian and military. These employment data types were found on two separate websites:

Civilian employment was obtained from the Office of Personnel Management's (OPM) FedScope website, which provides federal workforce data. From this link, one can click on "Employment" under the "Status Data" bullet point to access quarterly employment data cubes. Given that the report focuses on federal fiscal years, the employment data of interest is of Quarter 3 of a given year (i.e. September). When one clicks on a September cube of their desired year, one can filter the data by agency type ("Cabinet Level") and location ("United States" and a state of one's choice) in order to get the civilian employment numbers for the Departments of Defense (which is an aggregate of the Air Force, Army, Defense, and Navy cabinet agencies), Homeland Security, Veterans Affairs, and Energy. This data can be exported out as a PDF, CSV, or Excel file.

Military employment was obtained from the Defense Manpower Data Center (DMDC) website, which serves under the Office of the Secretary of Defense. From this link, one can hover over to DoD Data/Reports and select "Statistics & Reports". On this page, scroll to the "DoD Personnel, Workforce Reports & Publications" text, and click on its hyperlink. This opens up a new window

to access the military employment data. Similar to civilian employment, the data of interest is from September of a given year. One can scroll to the section "Military and Civilian Personnel by Service/Agency by State/Country (Updated Quarterly)", and download the Excel file for September of their desired year.

3.2 USASpending Data

This report utilized USAspending.gov for a large majority of the direct spending done by federal national security agencies in California. The spending types obtained from this site include contracts and grants for DoD, DHS, VA and DOE, as well as direct payments from VA (i.e. veterans' benefits). From this link, one can hover over to "Download" and click on "Custom Award Data". On this webpage, one can select the appropriate spending award types (contracts, grants, and/or direct payments), awarding agency (DoD, DHS, VA, DOE), recipient location (United States for "Country", and a specific state should you want to filter the data further), action date range (a fiscal year "FY 20XX"), and file format. This download process must be done once per agency.

For obtaining USAspending data, we have developed code that automates this entire process and easily captures all USAspending data in one instance. Please refer to our methods section of this process guide for more details.

3.3 Additional Data

3.3.1 SmartPay Data via FOIA

The remaining portion of direct spending detailed in these reports was SmartPay, a charge card program for federal employees. In order to obtain this data, Freedom of Information Act requests (commonly known as FOIAs) were filed to the federal national security agencies.

For more information about SmartPay, please visit the General Services Administration's (GSA) SmartPay website.

3.3.2 Data Obtained Online

A variety of data sources needed to be obtained online in order to assist with the data processing for this report. This includes:

- Federal Real Public Property data from the GSA. At this link, the GSA provides an Excel file that breaks down how much federal property is

spread out across the United States. This data is used to help localize the number of statewide DHS and VA civilian employees across counties and districts.

- American Community Survey data from the Census Bureau. At this link, type into the search bar at the center of the webpage "DP03". This returns a table of "Selected economic characteristics". From this table, one can filter to the appropriate geography (state, county, congressional district, etc.), year, and such to download into an Excel file. This data is used to help localize the number of statewide military personnel across counties and districts.
- NAICS to IMPLAN crosswalk from IMPLAN. At this link, IMPLAN provides a variety of crosswalks that one can use to relate data from other source to IMPLAN's 546 industries. On this page, one can go to the second heading titled "2017 NAICS to IMPLAN 546 Industries" and download the Excel file. This crosswalk is utilized to help relate the USAspending contracts data to IMPLAN sectors in order to run that spending data through the IMPLAN software.

3.3.3 Data Provided Raw/Self-Made

The repository contains some raw data files that we have prepared for this project. Specific instructions on how to access this repository and utilize these raw data sources are explained in the next section.

Chapter 4

Setting Up the Repository

Our code repository is necessary to repeat the California study. It may also be used as a frame work to run additional studies outside of the initial region. These instructions assume you will be using Github to obtain and manage a copy of the repository.

4.1 Obtain Repository

1. Using Github, navigate to the repository ([LINK TO REPO FOREVER HOME HERE]) and fork the repository to the user's account.
2. Clone the repository to a new RStudio Project.
3. Open the 'README.md' file ([LINK])

4. Fill out the parameters file with required specifications. The 'README.md' file contains general instructions, section 6.1 of the Methods section ([add link to section here]) has more detailed explanations.

4.2 Modify Data Files

Section used for instructing how to set up data repositories for custom user made data? Link back to this section from section 6

Chapter 5

Data Dictionary for Repository

This section details each file contained in the code repository and a summary of what it is used for in the project. For a more detailed explanation, please see the "Methods" section. [add link later]

5.1 MEIS_Methodology Folder [CHANGE NAME LATER]

This is the main folder, containing all data in the repository.

- **.gitignore:** Lists files that will be ignored when using Git to synchronize updates to repository.
- **.Rhistory:** Contains the history of commands given by a user in an R Session
- **LICENSE:** Contains licensing information for the repository
- **MEIS_Methodology.Rproj:** the R project package used to open the repository in an IDE such as RStudio.
- **README.md:** The first file you should read in any repository. Contains basic installation information, information on the repository contents and instructions on how to use the repository.
- **parameters.R:** Contains all variables that may require user editing in order for code to run properly. Use of this file is documented in the README.md file.
- **run_analysis_master.R:** The newest version of the code to process the Federal data. Currently this file is in development for the upcoming 2022 version of this project.

- **DEPRECATED_run_analysis_master.R:** This file contains the script used for the 2021 project. This process guide pertains mostly to the code running from this master script.

5.2 data Folder

This is a folder located in the main folder and is designed to hold data for this project.

5.2.1 raw Folder

This folder is located within the data folder and contains all raw data provided to the user. The purpose of the raw data is to simplify the data processing.

- **2007_to_2017_NAICS.xlsx:** This file contains the cross walk used in the master script to update 2007 NAICs codes to 2017 NAICs codes. This file is a new development to automate error checking when linking IMPLAN codes to NAICs codes. It is NOT used in the Deprecated master script.
- **2012_2017_NAICS_to_IMPLAN.xlsx:** This file contains the 2017 NAICS to IMPLAN crosswalk (available online from IMPLAN), as well as several additions. 2012 NAICS to IMPLAN values were appended to this file in addition to several 2002 NAICs codes that required updating. This file is a new development to automate error checking when linking IMPLAN codes to NAICs codes. It is NOT used in the Deprecated master script.
- **business_type_to_implan546_crosswalk.csv:** This file was created to serve as a crosswalk between grant recipient business types and their corresponding IMPLAN codes. This file was made by manually looking up each business type and entering its corresponding IMPLAN code value.
- **contract_recipient_to_district_crosswalk.xlxs:** This file was created to serve as an aid to error checking by assigning a District values to certain contract data that lacked them. Creation of this file involved looking up addresses of businesses with no given District in order to correctly assign a District to the data. It is NOT used in the Deprecated master script.

5.2.1.1 Blank_Sheets_for_R Folder

This folder contains the template spreadsheets used in the final preparation of data for entry into IMPLAN.

- **CommodityOutput2.xlsx:**
- **HouseholdSpendingChange4.xlsx:**
- **IndustrySpendingPattern5.xlsx:**
- **InstitutionSpendingPattern6.xlsx**
- **LaborIncomeChange3.xlsx:**

5.2.1.2 deprecated Folder

This folder contains raw data only used by the Deprecated master script.

- **2017_implan_online_naics_to_implan546.xlsx:** This file contains the crosswalk for linking 2017 IMPLAN codes to NAICs code.
- **2021_employment_totals.xlsx:**
- **DOD_County_Shares_R.xlsx:** This file contains the percentage of each County by surface area that is assigned to each Californian Congressional District as of 2021.

5.2.2 temp Folder

This folder is also located within the data folder and holds all files that are made while either master script is running. These files are deleted when the final output is finished being processed.

- **placeholderfortemp.txt:** This file prevents Github from deleting the temp folder, as it is empty by default.

5.3 output Folder

This folder is within the main folder and holds all of the final output files needed by IMPLAN in order to complete this project. New output files will get created and placed into this folder as scripts are run.

- **placeholder.txt:** This file prevents Github from deleting the output folder, as it is empty by default.

5.4 src Folder

This folder within the main folder contains all script files needed by the master script when it is running. Scripts in this folder may be used by either of the master scripts contained in this repository. DO NOT MODIFY ANY OF THESE FILES.

- **aggregate_usaspending.R:**
- **concatenate_usaspending.R:**
- **error_check_and_weight_contracts.R:**
- **error_check_grants.R:**
- **filter_usaspending.R:**
- **obtain_usaspending.R:**
- **split_usaspending.R:**

5.4.1 deprecated Folder

This folder within the src folder contains all scripts that are only used by the DEPRECATED_run_analysis_master.R script. DO NOT MODIFY ANY OF THESE FILES.

- **DEPRECATED_create_implan_sheets.R:**
- **DEPRECATED_error_check_contracts.R:**
- **DEPRECATED_error_check_grants.R:**

Chapter 6

Methods

The following section details how to use the data and R code provided as well as an explanation of how the code works.

6.1 Reviewing the Parameters File

The first step in working with this code involves looking at the Parameters file. This file allows for alterations in the data analysis based on particular criteria that one may want to modify in their study, such as geography of interest, year, agencies of interest, and so forth. The default entries in the parameters file are based on this study of California.

Readme will eventually link to this section

6.2 Making User Specific Data Files

The second step to focus on after modifying the parameters file is to ensure that user specific data files have been created in order to run this analysis. In this particular case, employment data (detailed in Section 3.1) and error check files were created in order to properly assign and allocate employment and spending data across California counties and congressional districts.

6.3 Processing the Data

After modifying the parameter files with the appropriate criteria and creating user specific data, one can begin working through the code

to process the data and prepare it for IMPLAN. Refer to the **deprecated_run_analysis_master.R** script to run through the data analysis process for this project.

6.3.1 Clearing Environment, and Loading in Packages and Parameters

The first lines of code provide some housekeeping steps prior to running through the analysis. Line 6 allows the user to clear their global environment in RStudio, removing all previously defined or loaded packages, variables, values, etc. In running through the analysis wholesale, a good first step is to clear one's environment. Following this is lines 9-14, which load in the various RStudio library packages that are needed to perform this analysis. For more information on these packages, please refer to our References section ("ADD IN LINK TO REFERENCES AND/OR R PACKAGE DESCRIPTIONS"). The final housekeeping step involves line 17, which loads in the parameters file mentioned above.

6.3.2 Loading in Functions

Lines 20-23 compose the second step of the data analysis process: loading in functions that standardize and simplify working through the data. Line 20 loads in a function that is used to place additional filters on the obtained USAspending data and write the filtered data into CSV files. These additional filters are also outlined in the parameters file, but the actual filter function is not run until lines 26-30 (see step 3 for more details).

Line 21 loads in a function that concatenates the USAspending contract and grant data files into one dataframe in R, while also removing direct payments from that concatenation (as it is calculated separate from contracts and grants. more on this will be detailed below). The actual concatenate function is not run until lines 39 and 41 (see step 5 for more details). Line 22 loads in a function that splits out the concatenated USAspending dataframe by agency, taking out DOE data from the main dataframe. This is done because DOE's spending activity has been kept separate from the main data analysis of the project. The actual split function is not run until lines 44-47 (see step 6 for more details). Line 23 loads in a function that aggregates the USAspending and DOEspending dataframes by IMPLAN sector. This gets the contracts and grants spending data for two IMPLAN activity sheets complete: one for the main statewide analysis, and one for the DOE statewide analysis. The actual aggregate function is not run until lines 55-60 (see step 7 for more details).

6.3.3 Obtaining the USAspending Data

Line 26 loads in a R script that performs a call to USAspending.gov's API that grabs the relevant contract, grant, and direct payment spending data that we need for our analysis and downloads it into two files (one for contracts, and one for grants/direct payments) at a specified file path location. In this case, the two CSV files obtained are downloaded into the data/temp folder. The parameters file is particularly important for this line of code, as the parameters file holds many of the filters that are used to sort what data is obtained from USAspending with this code.

6.3.4 Filtering the USAspending Data

Lines 29-34 begin the third step of filtering down the USAspending data obtained from the API call. Lines 29 and 30 read in the downloaded contracts and grants/direct payments files (respectively) and assign them each to a variable. Lines 33 and 34 perform the filter_usaspending function (that was loaded in at line 20) on each of those variables.

The filter_usaspending function has five elements that must be referenced: file_name, state, doe_filters, filters, and out_name.

- The **file_name** element is a call to whichever data file we want to pass through to be filtered - in this case, the two USAspending CSVs that are defined in lines 29 and 30.
- The **state** element is defined in our parameters, based on the state of interest for our study. In this case, that state is California. This element is added to ensure that the primary_place_of_performance_state is the same as the recipient_state (which was filtered in our obtain_usaspending function), thus ensuring that all data entries included in the analysis are in the state of interest for the study.
- The **doe_filters** element is defined in our parameters, based on **doe_offices**. This element is included to filter out DOE spending done in the state that is not national security-related. The basis for this filter element is from a list of DOE sub-agencies that were demarcated as national security-related.
- The **filters** element is defined in our parameters, based on **contract_columns** and **grant_columns**, respectively. This element is included to pare down the columns of data from the USAspending files to retain only the pertinent columns of data for our analysis.
- The **out_name** element is defined in our parameters, based on **c_out_name** and **g_out_name**, respectively. This element is included to give a naming convention for the files that get generated as a result of running the filter function.

After running the filter functions, two CSVs files should be generated in the data/temp folder, by the name defined from the **out_name** element.

6.3.5 Error checking the USAspending Data

Lines 37-38 begin the fourth step of checking for errors in the filtered USAspending data and fixing them to have an IMPLAN code associated with each entry. The process for error checking and fixing the data are relatively the same for contracts and grants - the only major difference being the variable of interest in fixing the errors. For contracts, the focus is on NAICS codes, while the grants look at business types.

6.3.5.1 Error Checking Contracts

Line 37 loads in a R script file that goes through the filtered USAspending contracts file and remediates issues in certain contract entries. The first lines in this R script load two files into two dataframes: 1) the filtered USAspending contracts file (**contracts**); and 2) a NAICS to IMPLAN crosswalk that was provided online on IMPLAN's website (**naics_to_implan**). After reading in the NAICS to IMPLAN crosswalk, there are a couple of code fixes and removal of duplicate codes that our team determined to, in our best knowledge, be the most accurate way to relate those NAICS codes to an IMPLAN code. After this, the NAICS to IMPLAN crosswalk is merged into the contracts dataframe by NAICS code in order to get an IMPLAN code associated with each contract entry.

However, this merge does not give all contract entries an IMPLAN code. This can be for a variety of reasons, including: 1) a contract entry from USAspending not having a NAICS code; and 2) an incorrect and/or outdated NAICS code that was not a part of IMPLAN's provided crosswalk. The lines of code following the merge work to fix the error contract entries. First, the contracts with no NAICS codes were hardcoded in the **contracts** dataframe to a specific IMPLAN code based on the contract recipient's name and the industry which they work in. Next, the contracts which had a mistyped or older NAICS code were pulled out to a new dataframe (**contracts_missing_implan**) and hardcoded to an IMPLAN code based on our team's best research into which IMPLAN code each entry would best fit into. This presents a degree of error when parsing out the contract spending, but also represents the best available workaround given time and information constraints. A new way of handling the error checks for contracts is detailed in Section 9: What's Next.

The final steps of this contracts error check involve dropping out the contract entries which had no IMPLAN code from the original **contracts** dataframe. Then, the dataframe which fixed the contracts that were missing IMPLAN codes (**contracts_missing_implan**) is merged back into the original **contracts**

dataframe. The **contracts** dataframe now has all contract entries "cleaned" with an IMPLAN code for each entry. Last, only the columns necessary for this analysis are selected, and then the cleaned **contracts** dataframe is written into a new CSV file.

6.3.5.2 Error Checking Grants/Direct Payments

Line 38 loads in a R script file that goes through the filtered USAspending grants and direct payments file and remediates issues in certain grants entries. First, the script loads in the filtered USAspending grants and direct payments file into a dataframe (**grants**). Next, we pull out the VA direct payments from the **grants** dataframe, and define it into its' own dataframe (**va_benefits**). This is done because VA direct payments does not a require an IMPLAN code and is thus calculated separate from contracts and grants from IMPLAN in the IMPLAN activity sheets. The **grants** dataframe filters out the direct payments based on the **assistance_code** column of the data, where entries that have an assistance code of 10 are the direct payment entries.

After VA direct payments are separate from grants, the next step is to load a business type to IMPLAN crosswalk into a dataframe (**business_to_implan**). This file was self-created by the team as the best method for relating grants data to IMPLAN codes, given that USAspending grants data entries do not provide for NAICS codes. The business type to crosswalk is then merged with the **grants** dataframe in order to get an IMPLAN code associated with every grant entry.

However, this merge does not give all grant entries an IMPLAN code. In the case of the grants data, they may be missing a business type value, or their business type may be one that is not captured in the self-created crosswalk. Thus, the lines after the merge work to hardcode IMPLAN codes into the **grants** dataframe based on the grant recipient's name and the industry which they work in. After these hardcodes, a second dataframe (**grants_missing_implan**) is defined to capture if any other grant entries are missing an IMPLAN code. If a grant entry is still missing an IMPLAN code, it would need to be fixed in the **grants_missing_implan** dataframe and then merged back into the original **grants** dataframe.

Now that the grants data has been "cleaned", the final steps in this script involve selecting only the necessary columns of data for the **va_benefits** and **grants** dataframes, and writing each one into their own respective CSV files.

6.3.6 Manually Fixing Congressional District Errors

Lines 40 and 42 provide an important notice to only continue with the code once some manual fixes have been implemented in the three cleaned CSV files produced from the above error check codes. **This step is essential**

to ensuring no weird errors come out along the way. The issue in the USAspending data for contracts, grants, and direct payments is the **recipient_congressional_district** column, where certain entries in all three CSV files have a district value of "NA" (contracts) or "90" (grants, and direct payments). There may be multiple reasons for this error, with the most plausible being that these data entries are in locations that span across multiple counties and/or congressional districts.

In order to properly remediate these issues, our team utilized the *DOD_County_Shares_R.xlsx* file in the raw data folder. Looking at this file, the "Districts" tab in that Excel sheet provides a breakdown of how much a county spans across one or multiple congressional district(s) based on land area from the California Redistricting Commission. We would then filter the contract, grant, or direct payment entries by county, and randomly assign a certain number of entries to a district based on their county. For example, if 20 contract entries that had a "NA" value in the district column were in Alameda County, we would assign 47% of those entries (~9) to CA-13, 42% (~8) to CA-15, and 12% (~3) to CA-17.

This method also inherently presents a degree of error with allocating spending across congressional districts. Even so, given time constraints, this workaround presented itself as the best solution at the time. Our team is working on a new way to address these manual fixes through code - for more details, refer to Section 9: What's Next.

6.3.7 Concatenating the USAspending Data

Lines 45-46

6.3.8 Splitting DOE from USAspending Data

Lines 49-52

6.3.9 Aggregating the spending Data for Statewide Numbers

Lines 58-59 and 61-63

6.3.10 Compiling Employment Data

Lines ??-??

6.3.11 Running For Loops to Generate IMPLAN Activity Sheets

Line ??

- Process data
  - Clean contracts and grant data-
  - Clean spending data
  - Error check contract spending data

  Will need to go into detail about changes in the code between this year (2021) and subsequent years

  More detailed mention of how the error checking of the USASpending.gov contract data is needed A detailed walk through of how to manually check data and use the multiple NAICS to IMPLAN crosswalks to catch data Mention how IMPLAN automatically removes any codes having to do with construction so those have to be manually coded

  Some errors occur due to the transaction not being given a NAICS code, those need to be manually fixed

  Issues occur with NAICS codes that apply to multiple IMPLAN codes-give detailed explanation of how this was worked around and data was processed and added back to the main cleaned data.

Chapter 7

Using IMPLAN

Once the IMPLAN activity sheets for your desired geographies have been produced, it is time to upload them into IMPLAN and allow the tool to run its analysis. When you log into IMPLAN, the homepage should display 3 boxes to navigate through: Regions, Impacts, and Projects.

- **Regions:** Click on Regions to begin your analysis. This should open up a map of the United States. Select the region of interest for the analysis (state, county, congressional district, etc.) and give this project a name. After saving the project, move on to the Impacts tab.
- **Impacts:** In the Impacts tab, look for the button to the right of the "Save" button. Click on that, and select "upload event template". From there, find the activity sheet for the geography of interest and select that for upload. Ensure that "Industry Change" and "Household Spending" can be selected, and choose those for uploading into IMPLAN. The data points for the respective IMPLAN sectors should then auto-populate on IMPLAN. If your analysis is including SmartPay spending data, be sure to manually add that in as type "Institutional Spending", specification 11001, and the value amount. Choose to select all events (the button to the left of the "Save" button) and drag them over to the geography's model on the right side of the IMPLAN screen. After that, select to run the model at the bottom right. Depending on how big the model, IMPLAN will take time to generate the results. Once the model is done with its analysis, click on the "View Results" button.
- **Results:** Once on this page, one can sort through the results of the IMPLAN activity sheet that was ran for a geography and download CSV files of a variety of data points. For this report, the data points of interest included economic indicators like output and full-time employment (FTE) opportunities, both generally and across industries, as well as government and tax revenue generated.

As part of our final step before developing and writing the reports, our team created code to combine all the input and output data from IMPLAN into one big spreadsheet which was used to create the graphs and charts seen throughout the reports.

Chapter 8

Conclusion/ Discussion

Hopefully we provide some guidelines and aid in discovering and processing this data so that quality studies can come about and Government spending can become more transparent.

Utilizing IMPLAN to manage government spending data doesn't come with a lot of instructions

You have to trust that the data entry is accurate, many instances of older NAICS codes exist and this has the potential to introduce errors into the code.

Limited time to process the data due to a new method of processing the data

Chapter 9

What's Next?

A section on where we hope to add and develop this process. Potentially the section to outline changes to the code we have already made for the upcoming 2021 report.

Chapter 10

License your GitBook

In the spirit of Open Science, it is good to think about making your course materials Open Source. That means that other people can use them. In principle, if you publish materials online without license information, you hold the copyright to those materials. If you want them to be Open Source, you must include a license. It is not always obvious what license to choose.

The Creative Commons licenses are typically suitable for course materials. This GitBook, for example, is licensed under CC-BY 4.0. That means you can use and remix it as you like, but you must credit the original source.

If your project is more focused on software or source code, consider using the GNU GPL v3 license instead.

You can find more information about the Creative Commons Licenses here. Specific licenses that might be useful are:

- CC0 ("No Rights Reserved"), everybody can do what they want with your work.
- CC-BY 4.0 ("Attribution"), everybody can do what they want with your work, but they must credit you. Note that this license may not be suitable for software or source code!

For compatibility between CC and GNU licenses, see this FAQ.

Bibliography

Ooms, J. (2014). The jsonlite package: A practical and consistent mapping between json data and r objects. *arXiv:1403.2805 [stat.CO]*.

Schauberger, P. and Walker, A. (2021). *openxlsx: Read, Write and Edit xlsx Files*. R package version 4.2.5.

Wickham, H. (2020). *httr: Tools for Working with URLs and HTTP*. R package version 1.4.2.

Wickham, H., Averick, M., Bryan, J., Chang, W., McGowan, L. D., François, R., Grolemund, G., Hayes, A., Henry, L., Hester, J., Kuhn, M., Pedersen, T. L., Miller, E., Bache, S. M., Müller, K., Ooms, J., Robinson, D., Seidel, D. P., Spinu, V., Takahashi, K., Vaughan, D., Wilke, C., Woo, K., and Yutani, H. (2019). Welcome to the tidyverse. *Journal of Open Source Software*, 4(43):1686.

Wickham, H. and Bryan, J. (2019). *readxl: Read Excel Files*. R package version 1.3.1.

Wickham, H., François, R., Henry, L., and Müller, K. (2021). *dplyr: A Grammar of Data Manipulation*. R package version 1.0.7.