California Military Economic Impact Study Process Guide

Britnee Pannell & Sumeet Bedi

Last Updated: 2022-02-10

Contents

Chapter 1

Introduction

The California Military Economic Impact Study series began in 2016, when the California Governor's Office of Planning & Research (OPR) and the Governor's Military Council requested the California Research Bureau, a unit of the California State Library, to conduct the first assessment. This study estimates detailed statewide and localized economic impacts of national security activity in California. The federal agencies identified as related to national security spending include the Departments of Defense, Homeland Security, Veterans Affairs, and specified sub-agencies of the Department of Energy. Economic activity detailed in the report 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 two reports, OPR secured grant funding from the U.S. Department of Defense to support two full-time, temporary positions to develop supplemental reports that localize the economic impacts detailed in the statewide report. In December 2020, the California Research Bureau 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, the Research Bureau completed the fourth edition 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, the Research Bureau was tasked with producing this process guide document.

This process guide and supporting documentation were developed 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, error checked, and analyzed in order to justify the estimates in our reports.

Chapter 2

Software Requirements

To recreate or replicate this study, specialized software is needed to obtain and process federal data. 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. Data can be broken into three types:

- **Employment**
- **USAspending**
- **Additional Data**

Please note, there is no guarantee that the data sources described below will exist in this form indefinitely.

3.1 Employment Data

This report focused on two types of employment data:

- **Civilian**
- **Military**

These employment data types were sourced from two separate websites.

Civilian employment is available from the Office of Personnel Management's (OPM) FedScope website, which provides federal workforce data. The report uses September data to align with the federal fiscal year. To obtain the data:

1. Load the Office of Personnel Management's (OPM) FedScope website.
2. Click on "Employment" under the "Status Data" bullet point to access quarterly employment data cubes.

3. Click September of the desired year.

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 is available from the Defense Manpower Data Center (DMDC) website. As with civilian employment, the report uses September data to align with the federal fiscal year. To obtain the data:

1. Load the Defense Manpower Data Center (DMDC) website.
2. Hover over to Department of Defense (DoD) Data/Reports and select "Statistics & Reports."

3. On the linked 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.

4. Scroll to the section "Military and Civilian Personnel by Service/Agency by State/Country (Updated Quarterly)" and download the Excel file for September of the desired year.

3.2 USASpending Data

This report utilizes USAspending.gov for the majority of its direct spending data. The spending types obtained from this site include contracts and grants for the U.S. Departments of Defense, Homeland Security, Veterans Affairs, and Energy, as well as direct payments from Veterans Affairs (i.e., veterans' benefits). Data is available from the "Custom Awards Data" webpage. Data is available by award type (contracts, grants, and/or direct payments), awarding agency (Defense, Homeland Security, Veterans Affairs, Energy), recipient location (United States for "Country," and, optionally, a specific state), action date range (a fiscal year "FY 20XX"), and file format. This download process must be done once per agency.

Code that automates this entire process has been developed to capture all USAspending data at once. Please refer to Section **6.3** in the Methods section for more details.

3.3 Additional Data

3.3.1 SmartPay Data via FOIA

The remainder of direct spending data used 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 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 are obtained online in order to assist with the data processing for this report. This includes:

- Federal Real Public Property data from the GSA. The GSA provides an Excel file that details federal property by location. This data is used to distribute statewide Homeland Security and Veterans Affairs civilian employment across counties and districts.
- American Community Survey data from the Census Bureau. Type "DP03" into the search bar at the center of the webpage. This returns a table of "Selected economic characteristics." From this table, filter to the appropriate geography (state, county, congressional district, etc.) and year to download an Excel file. This data is used to distribute statewide military employment across counties and districts. To address issues of small sample size, our team used five-year estimates of this data.
- NAICS to IMPLAN crosswalk from IMPLAN. IMPLAN provides a variety of crosswalks that are used to relate data from other sources to IMPLAN's 546 industries codes. On this page, 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 raw data files that have been built 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

The code repository is necessary to repeat the California study. It may also be used as a framework to produce estimates for other regions. These instructions assume GitHub will be used 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, and Section **6.1** of the Methods section has more detailed explanations.

4.2 Modify Data Files

There are certain custom user files that must be created as part of this analysis. In particular, the *TEMPLATE_emp.csv* file in the data/raw/ folder must be altered to properly load in and calculate employment data. Because employment data for this report is not easily obtainable or transmissible via code, our team has created this template file for users to fill in their statewide employment numbers. Also provided is the .csv file used in this study, labeled *CALIFORNIA_emp.csv*, to serve as an example of a properly filled out form. The first two columns in this file, active_duty_military and reserve_military, are military employment numbers from DMDC, while the remaining seven columns are civilian employment numbers (for the Departments of Defense, Homeland

Security, Veterans Affairs, and Energy) from FedScope. For more information on how to obtain this data, refer to Section **3.1** on data requirements for employment data. After edits are made to this file, it should be renamed based on the study area named in the "state" general variable in the parameters file and saved to the data/raw/ folder in the repository.

The repository requires two additional files to be obtained and added to the data/temp/ folder:

- **Federal Real Public Property Data**
- **American Community Survey Data**

Refer to Section **3.3.2** for details on how to retrieve these files. After these files are downloaded, they should be renamed and saved to the data/temp/ folder in the repository. It is recommended to use a consistent naming convention to make references and use of these files easier. Our naming convention involves naming the relevant data contained in the file, and the analysis year the file was obtained for. Hence the property data becomes *"fed_prop_2021.csv."* The ACS data was renamed *"2019_acs.xlsx"* to reflect the year the file was obtained for, and the data it contains.

The American Community Survey Data file requires an additional step of modification after saving. Within the file itself, the first row of data needs to be deleted. Then, the columns "Geographic Area Name" and "Estimate!!EMPLOYMENT STATUS!!Population 16 years and over!!In labor force!!Armed Forces" should be renamed to "geography" and "armed_forces_employed" respectively in order to simplify using the data in later steps.

The next section details data contained in the Repository and its purpose.

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 Section **6**, "Methods".

5.1 MEIS_Methodology Folder

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.
- **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.
- **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.
- **parameters.R:** Contains all variables that may require user editing for code to run properly. Use of this file is documented in the README.md file.
- **README.md:** The first file you should read in any repository. Contains basic installation instructions, information on the repository contents and guidance on how to use the repository.
- **.Rhistory:** Contains the history of commands given by a user in an R Session.

- **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 and is not yet ready for use.

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

- **2007_to_2017_NAICS.xlsx:** This file contains the crosswalk 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 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.
- **CALIFORNIA_emp.csv:** This file was created to hold employment numbers for selected federal agencies for the state of California. It serves as an example of how to fill out the **"TEMPLATE_emp.csv"** file and to allow for a repeat of the 2021 study.

- **contract_recipient_to_district_crosswalk.xlsx:** This file was created to serve as an aid to error checking by assigning a district value 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.

- **dod_county_shares.xlsx:** This file contains the percentage of each county by land area that is assigned to each California congressional district as of 2021.
- **TEMPLATE_emp.csv:** This file is a template for users to fill out employment numbers for their region of interest. Follow instructions in ([link to MODIFY DATA section]) to make use of this file.

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. For more information on the difference between the event types that comprise IMPLAN's activity sheet, refer to IMPLAN's article on this topic.

- **CommodityOutput2.xlsx:** An Excel sheet that goes into the IMPLAN activity sheets as the second worksheet tab.
- **LaborIncomeChange3.xlsx:** An Excel sheet that goes into the IMPLAN activity sheets as the third worksheet tab.
- **HouseholdSpendingChange4.xlsx:** An Excel sheet that goes into the IMPLAN activity sheets as the fourth worksheet tab. This Excel worksheet tab is what will hold the Veteran Affairs direct payments data.
- **IndustrySpendingPattern5.xlsx:** An Excel sheet that goes into the IMPLAN activity sheets as the fifth worksheet tab.
- **InstitutionSpendingPattern6.xlsx:** An Excel sheet that goes into the IMPLAN activity sheets as the sixth worksheet tab.

5.2.1.2 deprecated Folder {deprecated-raw}

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:** This file contains the necessary employment data broken out by each California county and congressional district

5.2.2 Temp Folder {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 located within the main folder and holds all the final output files needed by IMPLAN 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.

- **obtain_usaspending.R:** An R script that calls to USAspending's Application Programming Interface (API) in order to obtain the USAspending data desired based on the criteria defined in this R script.
- **filter_usaspending.R:** An R script that defines a function for further filtering the USAspending data that was obtained from the API call.
- **error_check_and_weight_contracts.R:** An R script that works through multiple steps and lines of code to fix errors in the USAspending contracts data based on the type of error.
- **error_check_grants.R:** An R script that works through multiple steps and lines of code to fix errors in the USAspending grants data.
- **concatenate_usaspending.R:** An R script that defines a function for concatenating a certain group of CSVs based on the file name pattern (in this case, the cleaned USAspending data) while ignoring the Veteran Affairs benefits CSV file.
- **split_usaspending.R:** An R script that defines a function for splitting out Department of Energy spending from the main USAspending data prior to aggregating the spending.
- **natsec_doe.R:** An R script that filters down the Department of Energy spending to national security-related offices and gives a national security adjustment proportion to use for Energy employment and SmartPay calculations.
- **aggregate_usaspending.R:** An R script that defines a function for aggregating a defined dataframe's spending figure (in this study, federal_action_obligation from USAspending data) by IMPLAN code.
- **generate_employment_dataframe.R:** An R script that calculates employment data and generates two dataframes, one for the county and one for the district IMPLAN sheets.

5.4.1 Deprecated Folder {deprecated-script}

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_error_check_contracts.R:** An R script of hard-code fixes to errors in the contracts data in order to get an IMPLAN code for every entry.
- **DEPRECATED_error_check_grants.R:** An R script of hardcode fixes to errors in the grants data in order to get an IMPLAN code for every entry.

- **DEPRECATED_create_implan_sheets.R:** An R script that runs two for loop codes that help to generate the IMPLAN activity sheets for every California county and congressional district.

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

This file allows for alterations in the data analysis that one may want to customize for their study, such as geography of interest, year, agencies of interest, and so forth. If any changes to the code are necessary in order to customize the output results, the only place users will need to make changes is in this parameters file. The default entries in the parameters file are based on the 2021 California study to provide an example of how to enter the desired variables.

6.1.1 Sections of the Parameters File

Each section of the parameters file defines variables used in a specific portion of the code. This section details the purpose of these variables and their intended use. The README.md file ([LINK]) contains correct syntax to use when making alterations to avoid breaking the code's functionality. Please refer to both sections as needed while altering the parameters file.

6.1.1.1 General Global Variables

This section contains general variables pertaining to the year the study takes place and the state the study focuses on.

- **f_year:** this variable represents the fiscal year, or the timeframe during which data was generated. Often, the year that the study data comes

from is older than the year the study is performed. This value is important to specify to ensure that 1) the data exists, and 2) that you are pulling the correct data.

- **year:** this variable represents the year that the study is performed in. Its main purpose is to ensure a naming convention is enforced for generated output files.
- **state:** this variable represents the state being analyzed. It is used to filter some data and to aid in enforcing naming conventions for generated output files.

6.1.1.2 User Generated Employment Files

This section contains the variables needed in order to generate user specific employment data pertaining to the study region.

- **res_mult:** This variable represents the multiplier required to calculate the number of equivalent full-time positions generated by members of the military reserves. In anticipation that this value may change over time, it is included in the parameters file.

- **national_sus_dhs:** This variable represents the number of Homeland Security employees that are "suppressed." This means that the precise location of their work is intentionally redacted for matters of national security. This number is updated quarterly on FedScope see Section **3.1**.
- **sus_dhs_mult:** This variable represents the percentage of suppressed employees in each state. This multiplier comes from 2016 data. End users should make their own approximations of this value if they are focused on a study region smaller than the state level.
- **acs:** This variable represents the file name used for downloaded American Community Survey (ACS) data used in calculating how military personnel are distributed throughout the study region (by county and congressional district).

- **dod_shares:** This variable represents the file name used for the DoD County shares file. In the event the user needs to use a different file than the one provided; the name of the new file can be altered here.

- **fed_prop:** This variable represents the file name of the document that stores information on federally owned buildings. The square footage of these buildings is used to estimate the proportion of federal employees in each agency per county and congressional district in the study area.

6.1.1.3 USAspending.gov API Variables

This section contains the variables needed to run the code to automate obtaining federal spending data from the USAspending.gov website API. Usaspending.gov provides additional documentation on how the USAspending API runs and its variable requirements.

- **agency_type:** This variable represents whether the agency or agencies in question awarded the contract money or funded it. USAspending.gov provides more information on agency type definitions.
- **agency_tier:** This variable represents which tier the agency resides in. An example of a top tier agency is the Department of Energy, and subtier of a top tier agency is the Missile Defense Agency (a subtier of the Department of Energy).
- **agency_name:** This variable represents the name of the agency or agencies targeted in the code.
- **tier_name:** This is an optional variable, representing the name of the agency only if it is a subtier agency.
- **date_type:** This variable represents the date type of the spending data. Whether the date range will focus on when the money was spent (action date) or the last time information was entered about a contract (last modified date).
- **date_range_start:** This variable represents the start date of the spending data being analyzed. It is usually the beginning of a fiscal year or quarter.
- **date_range_end:** This variable represents the end date of the spending data being analyzed. It is usually the end of a fiscal year or quarter.
- **awards:** This variable represents the award type of the spending data. It includes categories such as contract spending, grant award spending, Veterans Affairs disbursements and direct payments to individuals or businesses.
- **recipient_locations_country:** This variable represents the country to limit spending data to. USA needs to be specified to omit spending and resulting economic activity that occurs overseas.
- **recipient_locations_state:** This variable represents the state (if applicable) to limit the spending data to. Multiple states can be specified if put into a list format.
- **recipient_locations_county:** This optional variable represents the county(ies) to limit the spending data to, if further refinement within a state is desired. This variable can contain multiple counties for multiple states. However, this variable is mutually exclusive with the district variable. One or the other may be filled out, but NOT BOTH.

- **recipient_locations_district:** This variable represents the congressional district(s) to limit the spending data to. This variable can contain multiple districts for multiple states (although you may grab more data

than intended if the same district name is used in multiple states). Data is usually more accurate at the county level, and district data is often left blank where there are multiple recipients in a county that overlaps multiple districts. As a result, county-based analysis may be preferable.

6.1.1.4 Filter USAspending Variables

This section contains the variables required to filter the downloaded USAspending data.

- **doe:** this variable represents the syntax used in USAspending data for naming the Department of Energy (DOE). It is used to help filter and analyze DOE data.

- **contract_columns:** This variable represents the column headers of contract spending data to filter.
- **grant_columns:** This variable represents the column headers of grant data to filter.
- **c_label:** This variable represents a section of the name generated when the USAspending contracts data is automatically downloaded. It allows users to specify which unique portion of the file name the code searches for in order to simplify loading in the correct file for further processing.
- **g_label:** This variable represents a section of the name generated when the USAspending grants data is automatically downloaded. It allows users to specify which unique portion of the file name the code searches for in order to simplify loading in the correct file for further processing.
- **c_out_name:** This variable represents the desired output name of the filtered contracts data.
- **g_out_name:** This variable represents the desired output name of the filtered grants data.

6.1.1.5 Concatenate USAspending Variables

This section contains the variable required to run the code to merge the separate contracts and grants data into one file. This is to make subsequent portions of the code easier to run.

- **u_out_name:** This variable represents the desired output name of the concatenated USAspending data.

6.1.1.6 National Security DOE Spending Variables

This section contains the variable required to calculate which portion of federal DOE spending applies to national security.

- **doe_offices:** This variable represents the list of DOE sub tier agencies that receive national security funding.

6.1.1.7 Aggregate USAspending Variables

This section contains the variables required to aggregate the different spending data to prepare it for entry into Excel sheets and upload into IMPLAN.

- **u_state_outname:** This variable represents the desired output name of the file containing USAspending data that has been aggregated by IMPLAN code.
- **doe_state_outname:** This variable represents the desired output name of the file containing DOE data that has been aggregated by IMPLAN code.

6.2 Reviewing User Specific Files

After modifying the parameters file and before running any code, make sure you have obtained, modified and renamed the following data as per instructions in Section **4.2**. If these files are not properly named and located, the CODE WILL NOT RUN:

- **TEMPLATE_emp.csv:** Should be renamed based on the "state" parameter and saved to the data/raw/ folder.
- **Federal Real Public Property Data:** Should be renamed based on naming conventions and saved to the data/temp/ folder.
- **American Community Survey Data:** Should be renamed based on naming conventions and saved to the data/temp/ folder.

6.3 Processing the Data

After set-up is complete, code can be run to process data for loading into 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

Lines 1 through 17 in the deprecated master code serve to set up RStudio to process the federal data. This involves performing some housekeeping by clearing any variables already stored in the global environment, removing previously loaded packages, ensuring needed library packages are present and loading in the variables stored in the parameters file.

6.3.2 Loading in Functions

Lines 20-23 load in functions that standardize and simplify working through the data. Details about what purposes these functions provide and how they alter data will be described below in the relevant sections.

6.3.3 Obtaining the USAspending Data

Line 26 loads in a R script that performs a call to USAspending.gov's API to grab relevant contract, grant and direct payment spending data. The success of this script depends on correctly filling out the desired filter variables in the parameters file. For our study, we required data for the following awarding, top-tier agencies: Department of Defense, Department of Homeland Security, Veterans Affairs and Department of Energy, for the action date date-type from the 2020 fiscal year (October 1st, 2019 – September 30th, 2020) in the state of California in the United States. The obtain USAspending script downloads and unzips this data to the data/temp/ folder in the repository. The end output is in the form of two .csv files, one for contracts data and one for grant/direct payment data.

6.3.4 Filtering the USAspending Data

Lines 29-34 load in the data from USAspending.gov and use the *filter_usaspending* function to filter it. This serves to reduce the file size in order to speed up subsequent processing. Only data that was needed for this analysis was kept. All data was paired down to federal national security spending that occurred within the State of California using the *primary_place_of_performance* column.

For contracts data the following columns were kept: *federal_action_obligation, awarding_agency_name, awarding_sub_agency_name, award_description, funding_office_name, recipient_name, recipient_county_name, recipient_congressional_district, recipient_zip_4_code* and *naics_code.*

For grant/direct payment data the following columns were kept: *federal_action_obligation, awarding_agency_name, awarding_sub_agency_name, award_description, funding_office_name, recipient_name, recipient_county_name, recipient_congressional_district, recipient_zip_code, recipient_zip_last_4_code, assistance_type_code* and *business_types_description.*

6.3.5 Error checking the USAspending Data

Lines 37 and 38 begin checking for errors in the filtered USAspending data and modifying them to ensure they have an IMPLAN code for every entry. The only major difference between contracts and grants data is the variable referenced in fixing errors. Contracts data uses NAICS codes. Grants data uses business types.

6.3.5.1 Error Checking Contracts

Line 37 loads in a R script file that goes through the filtered USAspending contracts and remediates errors in certain contract entries. The first lines in the script load two files into dataframes: 1) the filtered USAspending contracts file (*contracts*); and 2) a NAICS to IMPLAN crosswalk provided by IMPLAN (*naics_to_implan*). There are several code fixes and removal of duplicate codes that our team determined, to the best of our knowledge, to be the most accurate way to relate a NAICS code to an IMPLAN code. After this, the crosswalk is merged into the *contracts* dataframe by NAICS code in order to get an IMPLAN code for 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 part of IMPLAN's provided crosswalk. The lines of code following the merge work to fix the contract entry errors. First, an IMPLAN code was hardcoded to the contracts with no NAICS code based on the contract recipient's name and industry sector. 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 **8**, "What's Next?"

The final steps of this script involve dropping the contract entries with no IMPLAN code from the original *contracts* dataframe. Then, the dataframe which fixed the contracts 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 the analysis moving forward are kept, and then the cleaned *contracts* dataframe is written into a new CSV file in the data/temp/ folder.

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 data to remediate issues in certain entries. First, the script loads in the filtered USAspending grants and direct payments file into a dataframe (*grants*). Next, Veterans Affairs (VA) direct payments are separated from the *grants* dataframe and placed into their own dataframe (*va_benefits*). This is done because VA direct payments are calculated separately from contracts and grants in the IMPLAN activity sheets. The *grants* dataframe filters out the direct payments based on the *assistance_code* column - entries with an assistance code of 10 are the VA direct payment entries.

After separating direct payments from grants, the next step is to load a business type to IMPLAN crosswalk into a dataframe (*business_to_implan*). This file was created internally as the best method for relating grants data to IMPLAN codes, as USAspending grants data did not have NAICS codes. The crosswalk is then merged with the *grants* dataframe in order to get an IMPLAN code for each 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 crosswalk. Thus, the lines after the merge hardcode IMPLAN codes into the *grants* dataframe based on the grant recipient's name and the industry sector. After these hardcodes, a second dataframe (*grants_missing_implan*) is defined to capture any other grant entries missing an IMPLAN code. If a grant entry is still missing an IMPLAN code, it will need to be manually 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 CSV files in the data/temp/ folder.

6.3.6 Manually Fixing Congressional District Errors

Lines 40 and 42 provide an important notice to manually fix some errors in the three files produced in the error checking step before running additional lines of code. **This step is essential to ensuring a reduction of errors further along in the analysis.** This step is needed because there are certain entries in all three CSV files where the *recipient_congressional_district column* has a district value of "NA" (contracts) or "90" (grants and direct payments). There

are multiple reasons for this error, with the most common being that these data entries are in counties that span across multiple congressional districts.

In order to properly remediate these issues, our team utilized the *dod_county_shares* Excel file in the data/raw/ folder. Looking at this file, the "Districts" tab in that Excel sheet provides a breakdown from the California Redistricting Commission of how much a county overlaps one or more congressional districts based on land area. Then the contract, grant, or direct payment CSV files' entries are filtered by county. These entries are randomly assigned to a district based on the percentage of their county that resides in each district. For example, if 20 contract entries without a district assignment were in Alameda County, 47% of those entries (~9) would be assigned to CA-13, 42% (~8) assigned to CA-15, and 12% (~3) assigned to CA-17.

This method inherently presents a degree of error with allocating spending across congressional districts. Given time constraints and data limitations, this workaround presented itself as the best solution at the time. Our team is working on a new and improved way to address these manual fixes - for more details, refer to Section **8**, "What's Next?".

6.3.7 Concatenating the USAspending Data

Lines 45-46 concatenate the cleaned USAspending contracts and grants data. The concatenate function works to bring together the cleaned USAspending contracts and grants files, while omitting the VA benefits file from that grouping. This concatenation step is necessary to eventually aggregate the contracts and grants spending by IMPLAN code for the activity sheets. Line 46 takes this concatenated USAspending contracts and grants dataframe (*concat_files*) and writes it into a CSV in the data/temp/ folder.

6.3.8 Splitting DOE from USAspending Data

Lines 49-52 separate the DOE spending from the main USAspending data. This is done because, to this point, our report has kept DOE data separate from the main analysis with the Departments of Defense, Homeland Security, and Veterans Affairs. Line 49 reads in the concatenated USAspending file that was generated from the previous step as a variable. Lines 51 and 52 perform the *split_usaspending* function on the concatenated data and define two new dataframes: *usaspending* and *doespending*. The *usaspending* dataframe will be used for the main analysis of this project, while the *doespending* dataframe will need some alterations prior to becoming a separate addition to the analysis of this project.

6.3.9 Filtering DOE to National Security-Related Data

Line 55 filters the *doespending* dataframe to only include national security-related activity. This step is important because, unlike the Departments of Defense, Homeland Security, and Veterans Affairs, only a subset of DOE activity is related to national security. A list of these sub-tier agencies can be found under the "doe_offices" variable in the parameters file. This new dataframe provides the DOE national security spending which will be used in IMPLAN. Additionally, a *doe_ns_adjustment* is calculated by dividing the sum of *federal_action_obligation* in *doe_ns_spending* by the sum of *federal_action_obligation* in the *doe_spending* dataframe. This adjustment calculation provides a rough proportion for how much DOE activity in the state is national security related. This value is used to assign statewide DOE employees and SmartPay accordingly.

6.3.10 Aggregating the Spending Data

Lines 59 and 60 perform the *statewide_aggregate* function for USAspending and DOE spending, pulling together all the spending data based on IMPLAN code. The CSV files which are written as a result of running the function provide the 'statewide spending data by IMPLAN sector' for our main analysis and DOE analysis.

Lines 62 through 64 do the simple calculation of aggregating the VA benefits data for the state (line 62), each county (line 63), and congressional district (line 64). The VA benefits figure from line 62 will go in the IMPLAN activity sheet with the statewide main analysis (i.e., the data and CSV from line 59), while the VA benefits data from lines 63 and 64 will figure into later code.

6.3.11 Compiling Employment Data

Line 67 loads in a R script that calculates the necessary employment data. All prior steps focused on preparing spending data for IMPLAN – this is the only step necessary for preparing employment data. The first couple of lines in this script read the state employment CSV (detailed in Section **4.2**) into a dataframe, *state_emp*, and define values for statewide military, civilian, Department of Defense, Department of Homeland Security, Veterans Affairs, and Department of Energy employment. From that point on, the code works to apportion the statewide employment numbers for each county and congressional district. Certain employment types have different methods and files needed in order to do this local apportionment.

ACS data is used for military employment calculations. We assume that the distribution of residents employed in the armed forces across counties and districts is a good proxy for estimating the location of their employment. The ACS

data file is read into a dataframe (*acs_data*) and two columns are added. The first column takes the number of employed armed forces in a county/district from the ACS and divides it by the statewide employed armed forces value to obtain a percentage, which we name *armed_forces_percent*. The second column, *military_personnel*, takes *armed_forces_percent* and multiplies it by the statewide military employment value (obtained from *state_emp*). Then, the geography and military_personnel columns are retained, and the county and district military employment numbers are saved into separate dataframes.

Next, the *dod_county_shares.xlsx* file (detailed in Section **5.2.1**) is used to apportion Department of Defense (DoD) civilian employees. ACS data does not account for the high concentration of DoD employees in Lassen County due to the presence of the Sierra Army Depot. Our self-created file, which is based on apportionment across counties from 2016 DoD employment numbers, does account for this, and is therefore used in place of the ACS armed forces apportionment. The *dod_county_shares.xlsx* file has two sheets, one for counties and one for districts. Each sheet is read into a dataframe to apportion the counties and districts, respectively. The counties dataframe (*dod_county*) apportions the statewide DoD civilian employees by the percent share that each county had for DoD employees in 2016. The districts dataframe (*dod_district*) is based off the counties dataframe but uses the percentage of county land area that lies within a district to distribute the DoD employees per county for each district. This distribution from county to district is then aggregated by district in order to get the DoD employees by district. Then, the two DoD dataframes (*dod_county* and *dod_district*) retain their respective geography column and DoD employment totals to complete the DoD apportionment.

GSA federal property data (detailed in Section **3.3.2**) is used to apportion Department of Homeland Security (DHS) and Veterans Affairs (VA) employment. The square footage of DHS and VA buildings across counties and districts in the state can provide a rough estimate for apportioning the statewide employment for those agencies. First, federal property data is read into a dataframe (*fedprop*), necessary columns are selected, and then appropriate filters are applied for property type and departments of interest. After this initial read in, a new dataframe is created just for VA properties (*va_fedprop*). A new column for employees is calculated by multiplying the statewide VA employment by the percentage of that region's building square footage divided by the state total. Then, two new dataframes are made to aggregate employees by county (*va_county*) and district (*va_district*). This entire process is then recreated for DHS, with three respective dataframes being made for the DHS calculations (*dhs_fedprop*, *dhs_county*, and *dhs_district*).

Next, the *county_emp* and *district_emp* dataframes are created. These dataframes will be used in the final step to help allocate the county and district employment for the IMPLAN activity sheets. Each dataframe merges together the four employment dataframes developed for each respective geography from the above steps (DHS, VA, DoD, and military) and ensures that all the

numbers in the dataframes are read as numeric rather than character strings. The final step for these two dataframes is to add in columns of implan_545 (which is just military employment) and implan_546 (which is the sum of DoD, DHS, and VA employment). For the *county_emp* dataframe, two additional columns are needed: *inverse_545* (the statewide military employees minus a county's military employees) and *inverse_546* (the statewide DoD, DHS, and VA employees minus a county's DoD, DHS, and VA employees). These new columns in these dataframes will be read in the next section and fill in the employment data for IMPLAN.

6.3.12 Running For Loops to Generate IMPLAN Activity Sheets

Line 70 loads in a R script that holds the entire process for generating IMPLAN activity sheets. The first lines of code in this script read in the individual Excel sheets from IMPLAN's activity sheet template and assigns them to a variable. These variables will be combined later in the process to generate one Excel file (the activity sheet for a given local geography), with each variable becoming its own sheet. After that, two variables are generated (*countynames* and *congressid*) by calling for a unique list of all county names and all congressional district numbers in the *usaspending* dataframe. From there, the localized geographies IMPLAN activity sheets are generated.

The first for loop code fills out and generates the county and inverse county IMPLAN activity sheets by iterating through the *usaspending*, *va_benefits_countiesagg*, and *county_emp dataframes*. Within the first portion of this loop, the *usaspending* dataframe is combined based on each county, and then aggregated by IMPLAN code. This data is combined with the employment data from *county_emp*, which is also combined based on each county. All of this data is stored into a variable, *temp*, which, along with some lines of code that format *temp*, constitute the first Excel sheet tab of the multi-tab IMPLAN activity sheet. Additionally, the *HouseholdSpendingChange4* variable is altered to input the respective county's *va_benefits_countiesagg* value. Then, a list (*templist*) is created, composed of the *temp* variable and all the Excel sheets that were read in. An output path and folder are defined, and the county activity sheet is written into this folder. The second portion of this loop mimics this entire process above but grabs the inverse of each county's data. In other words, the *tempooc* variable and everything that corresponds to it is based on all the spending and employment activity done outside of the county of interest. This data also gets written into an Excel file, which is named by the county followed by "in" to denote that it is a county's inverse IMPLAN activity sheet. For our analysis, this for loop code generates 116 total county IMPLAN activity sheets: 58 normal and 58 inverse.

The second for loop fills out and generates the district IMPLAN activity sheets by iterating through *usaspending*, as well as the *va_benefits_districtsagg* and

district_emp dataframes. Due to limitations with IMPLAN's software, the district inverse models are not run; hence, the district inverse activity sheets are not needed. This for loop follows the exact same process as the first portion of the county for loop – the only difference being that the data is aggregated by district, and that different variable names are utilized (i.e., *temp2*). This for loop does denote a separate output folder to store all the district IMPLAN activity sheets, and names each Excel file based on "CA-" followed by the district number. For our analysis, this for loop code generates 53 total district IMPLAN activity sheets.

6.4 Using IMPLAN

At this point, the activity sheets for the desired local geographies have been generated and are ready to be run through IMPLAN. Note that the data for the statewide main analysis and statewide DOE analysis needs to be written to a file, and copied over into IMPLAN's activity sheet template, in order to run the statewide analysis. For more details on how to run these activity sheets through IMPLAN, refer to their guide on how to use the activity sheets in their software.

Chapter 7

Conclusion/ Discussion

From our analysis for this project, we found that national security contributes significantly to California's economy. The total impact appears similar to high profile sectors such as the agriculture and film industries. In 2019, the federal government invested at least $47.0 billion and directly employed approximately 348,000 residents in the state. This resulted in $181.2 billion in economic impact and supported over 792,000 full-time equivalent jobs in California.

Our process for this project provided us with some lessons:

- FOIA data can be difficult to obtain, as it is dependent on how quickly and accurately a federal agency responds to your request. From our experience, we have received responses to our FOIA requests at least six months after our initial submission.However, data obtained from FOIAs (SmartPay) constituted 3.4% of the nation's national security-related direct spending. Depending on your resources and priorities, choosing to omit this data would typically have minimal impacts on the overall estimates.
- Manually obtaining and processing government spending data (such as USAspending) is time and labor intensive. In developing an automated way of making this more efficient, we found that there were not a lot of good resources on how to interact with APIs for the initial data grab. The APIs that exist do not have clear instructions, and it takes time to learn how to interact with them. Even in automating this process, the limitation of time and labor presents itself. As a caution, take ample time to understand the APIs to ensure the correct data is obtained, especially if branching out from the API used for this study.

- Within the manually processing of government spending data, a particular pain point was the remediation of errors in the USAspending data. Contracts data may have had old, mistyped, or missing NAICS codes that required additional searching of previous crosswalks to properly allocate

spending entries to the correct IMPLAN code. Additionally, the errors of missing or "90" congressional district values require diligence in approach to remediate that issue.
- Processing this data in IMPLAN for the county and congressional district models can be time consuming, and approximate time will vary based on the number of counties and congressional districts being processed.

For researchers looking to perform this study on their own, we hope to offer some advice on the planning and timeline stages for such a project:

- If it is determined that FOIA data is wanted and/or needed for your study, file your requests many months in advance.
- Running the code up to the aggregate function (see Section **6.3.10**), and looking at how many IMPLAN sectors have a spending amount should provide a good barometer for how long the IMPLAN processing will take for the statewide model. The California statewide model had 400 IMPLAN sectors and took roughly 20 minutes to run.
- After running the for-loop code (Section **6.3.12**), for generating localized IMPLAN activity sheets, note that the activity sheets for the multi-region inverse models will take significantly more time than the sheets for the normal models. Roughly speaking, a single inverse model takes a couple of hours to finish running, while a single region model should complete within half an hour.

The process guide for this study was created to be a useful tool for researchers to conduct similar economic impact studies. Through this document, we hope that additional quality studies continue to come about and assist in making government spending data more readily accessible and transparent. Furthermore, we hope to continue making process improvements in order to ensure that this study retains its functionality and capacity for future years.

Chapter 8

What's Next?

Looking past this process guide, our team is working to continue improving tasks that make up this project. Moving forward, here are some changes and modifications we hope to implement to improve the process outlined above:

- Making the entire process for this analysis applicable to multiple states. While our team has worked to do this by thoroughly developing our parameters file, we recognize that certain parts of this process are inevitably California-centric. For example, some of the data files used in this report (such as the *dod_county_shares.xlsx* file) represent a fix only applicable to California. Our team will consider future process improvements on this point that can make the analysis process even more broadly applicable.
- Being vigilant of more APIs so that more of the data analysis process can be automated. One of the biggest breakthroughs for this analysis has been developing a code to automate downloading data from USAspending.gov. However, most of the data sources used for this report (such as FedScope or DMDC) do not have clearly defined or available APIs. Our hope is that these websites will continue to develop ways that can make obtaining data easier, and that we will be able to document, code and share that process.
- Automating the error checking portion of the code. Our current process of manually checking for errors across the USAspending data sources is time and labor intensive. Our team hopes to generate a more streamlined process that harnesses the power of code to lessen the amount of manual time and labor needed to remediate errors from USAspending.
- Implementing small area estimation in our calculations for county and district apportionment of spending and employment data. Small area estimation provides a more accurate statistical basis for parsing out statewide numbers across smaller local geographies, such as counties and districts. Our team hopes to spend some time learning more about small area es-

timation in order to more robustly apportion data across counties and districts.

- Keeping the process up to date with inevitable changes. Of particular concern are future updates to NAICS codes and state's congressional districts (due in 2022). These changes in code and geography will certainly affect the analysis, and it will be prudent to continue updating and developing our code and documentation to handle these changes.

Ultimately, we hope to develop a one size fits most method for obtaining and analyzing the impacts of federal spending within the United States.

Chapter 9

License

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.