**Microsimulation**

**Model Documentation **

First Draft

November 2, 2018



**Microsimulation**

**Model Documentation for R Implementation**

November 2, 2018

CONTENTS

[Chapter 1. Introduction 6](#_Toc529203697)

[1.1. Purpose of Report 6](#_Toc529203698)

[1.2 Model Overview 6](#_Toc529203699)

[1.3 Structure of Report 6](#_Toc529203700)

[Chapter 2. Model Overview 7](#_Toc529203701)

[2.1 Model Purpose 7](#_Toc529203702)

[2.1.1 Microsimulation Model 7](#_Toc529203703)

[2.1.2 Benefit Financing Module 7](#_Toc529203704)

[2.2 Model Inputs 7](#_Toc529203705)

[2.2.1 FMLA Survey 7](#_Toc529203706)

[2.2.2 American Community Survey 7](#_Toc529203707)

[2.2.3 Current Population Survey 7](#_Toc529203708)

[2.2.4 User-Defined Inputs 7](#_Toc529203709)

[2.3 Model Outputs 7](#_Toc529203710)

[2.4 User Interface 7](#_Toc529203711)

[Chapter 3. R Implementation 8](#_Toc529203712)

[3.1 Overview of Code Structure 8](#_Toc529203713)

[3.2 Main Simulation Function 8](#_Toc529203714)

[3.2.1 Preprocessing 9](#_Toc529203715)

[3.2.2 CPS Imputation 9](#_Toc529203716)

[3.2.3 FMLA Imputation 9](#_Toc529203717)

[3.2.4 ACS Simulation 10](#_Toc529203718)

[3.3 Pre-processing Functions 10](#_Toc529203719)

[3.4 Imputation Functions 10](#_Toc529203720)

[3.4.1 CPS Imputation 11](#_Toc529203721)

[3.4.2 FMLA Imputation 12](#_Toc529203722)

[3.4.2 ACS Imputation 12](#_Toc529203723)

[3.5 Simulation Functions 12](#_Toc529203724)

[3.5.1 Logistic Regression 12](#_Toc529203725)

[3.5.2 K-Nearest Neighbors 12](#_Toc529203726)

[3.5.3 Random Forest 12](#_Toc529203727)

[3.6 GUI functionality 12](#_Toc529203728)

[3.7 File Summary 12](#_Toc529203729)

[Chapter 4. Python Implementation 14](#_Toc529203730)

[3.1 Overview of Code Structure 14](#_Toc529203731)

[3.2 Main Simulation Function 14](#_Toc529203732)

[3.3 Pre-processing Functions 14](#_Toc529203733)

[3.4 Imputation Functions 14](#_Toc529203734)

[3.4.1 CPS Imputation 14](#_Toc529203735)

[3.4.2 FMLA Imputation 14](#_Toc529203736)

[3.4.2 ACS Imputation 14](#_Toc529203737)

[3.5 Simulation Functions 14](#_Toc529203738)

[3.5.1 Logistic Regression 14](#_Toc529203739)

[3.5.2 K-Nearest Neighbors 14](#_Toc529203740)

[3.5.3 Random Forest 14](#_Toc529203741)

[3.6 GUI functionality 14](#_Toc529203742)

[Chapter 5. Algorithms 15](#_Toc529203743)

[5.1 Logistic Regression 15](#_Toc529203744)

[5.1.1 R Implementation 15](#_Toc529203745)

[5.1.2 Python Implementation 15](#_Toc529203746)

[5.2 K-Nearest Neighbor 15](#_Toc529203747)

[Chapter 6. Validation 16](#_Toc529203748)

[6.1 FMLA Internal Validation 16](#_Toc529203749)

[6.2 Robustness Checking 16](#_Toc529203750)

[6.3 Sensitivity Checks 16](#_Toc529203751)

[6.4 Standard Error Estimation 16](#_Toc529203752)

[Chapter 7. Conclusion 17](#_Toc529203753)

[Bibliography 18](#_Toc529203754)

**Table of Exhibits**

[Exhibit 1: Overview of Model 6](#_Toc528747820)

[Exhibit 2: Overview of R Code Structure 8](#_Toc528747821)

[Exhibit 3: Summary of inputs to policy\_simulation() function 8](#_Toc528747822)

[Exhibit 4: Pseudocode for K-Nearest Neighbor 11](#_Toc528747823)

# Chapter 1. Introduction

## 1.1. Purpose of Report

This document describes in detail the implementation of the IMPAQ-DOL worker leave microsimulation model. It describes the purpose and functionality of the model from a high level, as well as detailing the technical methodology. Further included are instructions on the set up and use of the microsimulation model for typical users, as well as finer configuration options for advanced users seeking additional customization.

## 1.2 Motivation and Background

Nearly every developed country in the world has a public paid maternity leave available for its workers. However, the United States remains an outlier; there is no federal requirement for employers to offer paid leave or sick days to employees. As a result, access to leave is scant among the American workforce. In 2016, only 14 percent of all US workers have access to paid family leave through their employers, and 68 percent have paid sick leave.[[1]](#footnote-2) Absent federal policy, Some states and municipalities have moved forward on paid family leave. California enacted paid family leave legislation in 2002, New Jersey in 2008, Rhode Island in 2013, New York in 2016 (effective January 2018), the District of Columbia in 2017 (effective July 2020), and Washington in 2017 (effective January 2020). Other states and municipalities have shown interest in the feasibility of adopting their own paid-leave policy.[[2]](#footnote-3) However, one often-cited obstacle to providing paid family and medical leave in the US is the anticipated cost. Policymakers need to ensure the program will be financially sustainable.

Estimating cost of state and local paid leave programs are not straightforward due to limitations of current data sources. The lion’s share of program costs are wage replacement benefits paid out to eligible leave takers. To properly estimate this, policymakers must estimate the number of eligible workers within their constituency is and their leave taking behavior. The best data source available for estimating leave taking behavior of the US population is USDOL’s Family and Medical Leave survey (referred to as the FMLA survey).[[3]](#footnote-4) The survey captures in great detail the leave taking behavior and demographic characteristics from a nationally representative sample. However, with only 2852 respondents, insufficient sample size and respondent privacy become barriers to employing traditional data analysis techniques at the state and local level to obtain these estimates.

Microsimulation methods overcome these barriers through imputing leave taking behavior observed in the FMLA data to the larger, more robust American Community Survey (ACS) from the Census Bureau. The modeling approach relies on the significant overlap of demographic characteristics (such as age, sex, and race) collected in both the FMLA and ACS surveys that are related to leave taking behavior. The associations between these characteristics and behavior in FMLA data have been fit to ACS data via logistic regressions in previous microsimulation models. [Research on ACM and other previous models]. With a larger and more robust sample, ACS is then able to more precisely estimate leave taking behavior at a national or state level.

## 1.3 Model Overview

The IMPAQ-DOL model is a robust, accessible tool to assist in the design of paid leave programs and research of leave taking behavior at the national and state level. The simulation model proceeds in six broad steps as indicated in Exhibit 1. First, the input data sets are individually cleaned and prepared for use in the model. Second, FMLA data is used to calibrate the leave taking estimation model for application in ACS data. Third, leave taking behavior is imputed on an ACS data set using the estimation model. The ACS data is selected based on the user-defined geography of interest; be it national-level leave taking, or leave taking for a specific state. Fourth, participation and benefits received are calculated in the ACS using user-specified leave program characteristics and behavioral assumptions. Fifth, if the user has elected to calculate what tax structure will be required to pay for the program, the benefit financing module calculations are run. Finally, the model displays simulation results and financing estimates in tabular and graphical form.

Exhibit 1: Overview of Model

6. Output simulation results and estimates

3. Impute leave taking for ACS data

4. Adjust ACS data based on the characteristics of the simulated leave program

1. Clean Census & FMLA input data sets

2. Calibrate leave taking estimation model from FMLA data

5. Apply benefit financing module calculations

## 

## 1.4 Structure of Report

This report proceeds as follows. Chapter 2 presents a high level overview of the microsimulation model by describing its purpose, its main components and its logical flow. This chapter is aimed at guiding less technical users who may not be familiar with the intricacies of microsimulation modeling and R programming. Chapter 3 describes the R implementation, detailing the primary functions and interconnectedness of the various files and functionalities. This chapter is aimed at the technical programmer familiar with modelling in R.

# Chapter 2. Model Overview

This chapter is intended to provide enough detail for the average user to understand the broad picture of what the model intends to accomplish, what the model’s inputs and outputs are, and how the average user can specify and run a leave-taking simulation. This chapter focuses mostly on the big picture of what the model intends to accomplish, and how the user can utilize the Graphical User Interface (GUI) to customize this. Detailed technical discussions of internal, back-end components are reserved for Chapters X-X.

## 2.1 Model Purpose

The primary purpose of this model is to provide a robust, accessible tool to assist in the design of paid leave programs and research of leave taking behavior at the national and state level. There are two distinct components of this model; the microsimulation module and the benefit financing module.

### 2.1.1 Microsimulation Module

This microsimulation module’s primary purpose is to provide accurate estimates of leave taking behavior and leave program participation for both the US and individual states. To facilitate broader use of this model compared to predecessors, we designed the model and its results to be accessible, flexible, and transparent to a non-technical audience. Our focus in development was also to on the technical performance of the model. We have built and tested a number of behavioral estimation methods to establish the best way(s) to accurately perform leave taking estimation.

### 2.1.2 Benefit Financing Module

To start a paid leave program, a state needs to know not only what the program will cost, but what tax structure or financing plan will be adequate to cover the program’s costs. After the microsimulation model produces the amount of estimated benefit payouts, the benefit financing module helps the user obtain this. As inputs, this module uses the same ACS data and user inputs on a theoretical tax structure to simulate. The module outputs the simulated annual revenue from the tax structure, and compares it with the annual amount of benefits that will be paid out.

While bulk of program costs are leave benefit payouts, there are administrative and procedural costs to running a paid leave program. For example, staff time must be spent validating eligibility, checking for improper payments, and investigating evidence of fraud. The benefit financing module also includes a tool to calculate these kinds of costs based on user inputs.

Altogether, the benefit financing model allows users to easily take the output from the microsimulation model and come up with a tax structure or other financing plan to cover the costs of the theoretical leave program in question.

[ABF team to elaborate]

## 2.2 Model Inputs

### 2.2.1 2012 FMLA Survey

USDOL’s 2012 FMLA survey behavior is the third wave of a cross-sectional survey on paid leave.[[4]](#footnote-5) Respondents are asked about leave taking behavior in great detail, including: the number, lengths, and types of leaves taken, to what extent the employer provided pay while on leave, and whether or not some or additional pay while on leave would alter their leave-taking behavior. This survey’s data is this model’s primary data source for leave taking behavior in the US.

The survey interviewed 2852 employees, of which 1551 responded that they took or needed to take leave in the past 18 months. These 1551 respondents provided details on the leave(s) they took or needed to take. State of residence is not available in the survey due to risk of personal identification. As a result, this data alone cannot inform state-level estimates of leave taking.

### 2.2.2 American Community Survey

The American Community Survey (ACS) is a large national representative sample of individuals within the US. The ACS is conducted on a continuous basis, but public use ACS data is released on an annual basis in 1-year and 5-year data sets.

This model uses 2012-2016 ACS 5-year data (referred to as ACS data onwards) to maximize sample size and to coincide with the 2012 FMLA survey. This adds an implicit assumption to our model’s estimates: that 2012 status-quo leave taking behavior did not change significantly over the 2012-2016 time period. To simulate leave taking in a single state, the model filters the national ACS dataset to only include observations from the given state. To simulate nation-wide leave taking, no such filter is applied and the entire ACS data set is used.

In total, the entire ACS data set contains 12.9 million observations, and individual states range from approximately 20,000 (Wyoming) to 1.1 million (California) observations.

### 2.2.3 Current Population Survey

While the ACS has a very rich set of variables, there are a handful of variables the model requires that ACS does not contain or contains insufficient detail on. These variables are: whether pay is received on an hourly basis; employer size; the number of employers that the person worked for in the last 12 months; and weeks worked in the last 12 months.

To estimate these, we use the Current Population Survey (CPS). The CPS is another nationally representative Census survey that does contain the required variables in sufficient detail. We follow the ACM model’s method of imputing these values via logistic or ordinal logistic regression from the CPS to the ACS on a set of overlapping demographic variables. [may need to change this language if we make this imputation modular]

### 2.2.4 User-Defined Inputs

Prior to executing the simulation, the model allows the user many different options to modify the simulation through a graphical user interface (GUI). The GUI user inputs fall into three main categories:

* **Program inputs:** Inputs that define the characteristics, rules, and benefits of the leave program to be simulated. (e.g. weekly program benefits paid, and maximum length in weeks
* **Behavior inputs:** Inputs that define the assumptions to be used for simulating the population’s behavioral response to the presence of a leave program (if one is specified).
* **Advanced inputs:** Other **i**nputs the average user is not expected to use often, but advanced users may wish to utilize.

## 2.3 Model Outputs

The model’s output is at its core a modified version of the ACS data file. The original ACS data file is essentially a very large spreadsheet, with each row representing an individual in the ACS and each column a different characteristic variable. The simulation takes this file, and adds several columns to represent leave taking and program participation behavior. From there, the model computes the summary data necessary to create the charts and tables shown in the GUI. Exhibit 3 is a visualization of the model’s output data set and Exhibit 4 is an example graph produced by the GUI from the model output data set.

Exhibit 2. Simulation Output Dataset Visualization [Need a more polished version of this]

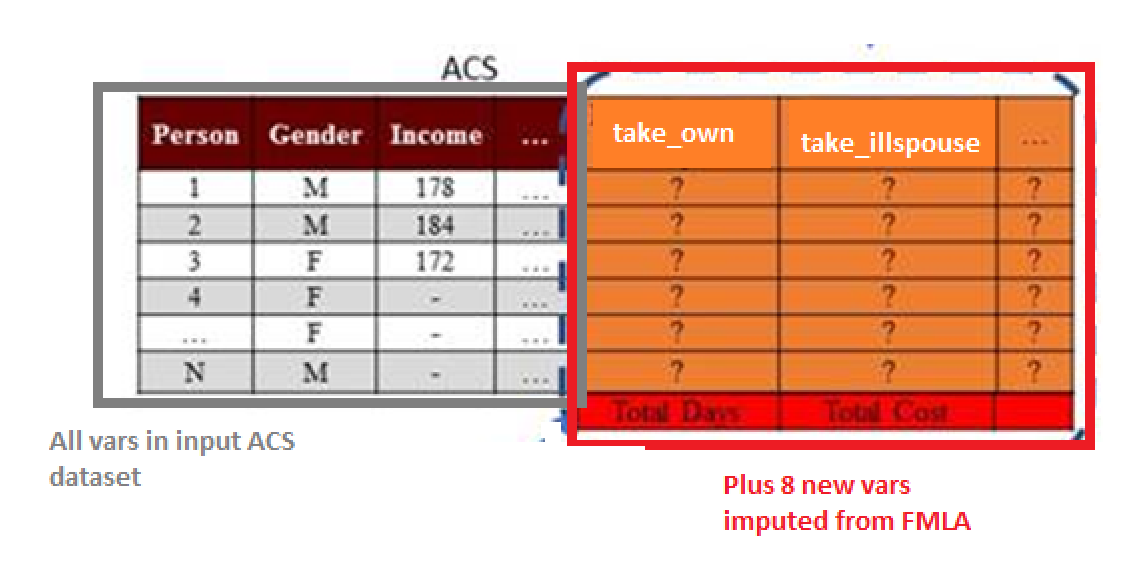
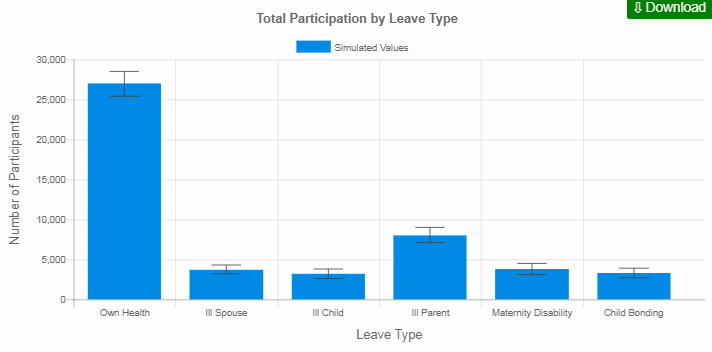


Exhibit 3. Simulation Output Graph Example [Need a more polished version of this]



[Description of ABF outputs]

## 2.4 User Interface

The actual implementation of the Graphical User Interface (GUI) from the perspective of source code is described in Chapter X. Once the user submits their desired inputs and the simulation completes execution, the GUI also displays the output results of the simulation. User parameters are divided among three different tabs, each corresponding to the types of parameters mentioned in the previous section: Program inputs, Behavior inputs, and Advanced inputs. Exhibit 2 is a sample screen shot of this GUI. A full description of the GUI and each of its field can be found in Appendix A.

Exhibit 4: Sample Screenshot of GUI [needs to be updated]



The outputs displayed by default in the GUI are [elaborate on GUI output displays once finalized].

# Chapter 3. R Implementation

This chapter describes in detail how the microsimulation model is implemented in the R programming language. It describes the purpose and interaction of the functions, variables, and files that comprise the model. This chapter is designed to provide a developer with the full information required to edit the underlying source code of each stage of the model. Section 3.1 provides an overview of the code’s structure and execution. [Complete summary of chapter structure]

## 3.1 Overview of Code Structure

At a very high level, the model performs two key functions.

1. The first is to infer the unobserved leave taking behavior of individuals within the ACS based on the leave taking behavior of comparable individuals within the FMLA. Because the sample size of the ACS is much larger than the FMLA, accurate inference of leave behavior within the ACS allows for a broader demographic and geographic range of leave analysis within the United States. This “inference component” of the microsimulation is purely statistical, in the sense that it is purely an imputation procedure.
2. The second key function of the model is to simulate counterfactual leave-taking behavior in the presence of different leave taking programs. This component relies on behavioral assumptions regarding how individuals are likely to respond to changes in the “leave taking environment.”

The general R implementation is summarized in Exhibit X. The user defines the state for whose population the model will simulate leave taking for. The model uses the observations in the ACS of individuals from the corresponding state, which together with the FMLA survey form the main input data sets for the model. The user also specifies values for a variety of different parameters. These parameter values mostly represent one of two adjustments to default simulation settings: 1) adjustments the rules, limits, and restrictions governing participation and benefit collection for the simulated paid, or 2) adjustments to the population’s leave-taking behavior.

**Exhibit 5: Overview of R Code Structure**

**Input**:

ACS, FMLA Data

0\_master\_execution\_function.R

policy\_simulation()

1\_cleaning\_functions.R

2\_pre\_impute\_functions.R

3\_impute\_functions.R

4\_post\_impute\_functions.R

policy\_simulation()

subfunctions

**Input**:

User-specified simulation parameters

**Output**:

ACS data set simulated leave taking and program participation behavior

**Additional Output**:

Leave taking and program participation summary statistics

5\_output\_analysis\_functions.R

The 0\_master\_execution\_function.R file contains the policy\_simulation() function, which governs the entire simulation. The function receives the input data and parameters, which are then fed to their appropriate sub-functions, each of which performs a number of different operations. Those subfunctions are contained within separate files, which group them into four rough categories:

* 1\_cleaning\_functions.R contains functions to load the input CPS/ACS/FMLA data sets, create the necessary variables for the simulation, and save cleaned versions of the datasets for use in the simulation.
* 2\_pre\_impute\_functions.R contains functions to prepare the FMLA data set for imputation into ACS, and executing the CPS imputation into ACS. These functions differ from the cleaning functions above in that user parameters can modify how these functions work, whereas no user parameters modify how cleaning functions operate.
* 3\_impute\_functions.R contains functions to impute leave taking and other necessary variables from the FMLA data set into ACS. Functions defining a wide variety of different imputation methods are included in this file, allowing the user to specify which method to use.
* 4\_post\_impute\_functions.R contains functions that execute post-imputation leave taking behavior and program participation adjustments, and produce the completed, final output ACS data set.

The final output of the simulation model is a modified version of the input ACS data set, with additional variables added to represent leave-taking and program participation behavior. The final .R file, 5\_output\_analysis\_functions.R, defines functions that analyze the output ACS data set and produces statistics, charts, and tables summarizing resulting leave taking behavior, program participation, and benefits distributed for the simulated population.

The basic steps for the user to execute the simulation are illustrated in Exhibit X. An example execution file is demonstrated in the TEST\_execution.R file included with this model.

Exhibit 6. R Model Execution Diagram

User runs 0\_master\_execution\_function.R, which defines the policy\_simulation() function in the user’s R environment

policy\_simulation() runs the following files to define its sub-functions:

1\_cleaning\_functions.R

2\_pre\_impute\_functions.R

3\_impute\_functions.R

4\_post\_impute\_functions.R

5\_output\_analysis\_functions.R

User’s R environment can now execute policy\_simulation() with user-specified parameters

The model generally follows the structure of the original Albelda/Clayton-Matthews microsimulation model (ACM)[[5]](#footnote-6) with two major exceptions:

* The ACM model hard-codes a probabilistic logit model to impute variables in ACS using data from FMLA. We have built in a modular imputation structure that allows for different imputation methods and different independent variables to be used for this imputation.
* We concentrated on the simulation and analysis of a more focused set of outcomes; primarily population-level leave taking behavior, participation in the simulated leave program, and simulated benefits paid out. This was done with the intent to make the model as useful as possible to state policymakers wishing to estimate what it would take to finance a certain paid leave program. Conversely, the model output is also intended to help policymakers understand what kind of a paid leave program they could offer conditional on a fixed budget level.

## 3.X Background on Input Datasets

Chapter 2 provided a high-level overview of the three input data sets (ACS, FMLA, and CPS). However, there are several nuances to these data sets that are relevant to understanding many of the structural details and development decisions of the model. These are detailed below.

### 3.X.1 FMLA Survey Background

The FMLA survey is the best available data source for nationally-representative leave-taking behavior. However, it has a number of design decisions that affect its use in our model.

***Categorizing Leave Taking Status.*** The screening questions of the survey first establish whether the respondent is in one of four categories of leave taking behavior within the prior 18 months, which dictated the sections of the survey a respondent received. All respondents received sections D (demographics questions) and E (employment questions).

1. Leave taker only (Received section A – leave taking questions)
2. Leave needer only (Received section B – leave needing questions)
3. Dual taker and needer (Received both section A and B)
4. Employed, but did not need nor take leave. (Received section C – just a single question confirming no leave taken/needed)

***Categorizing Type of Leave.*** Many components of the model are disaggregated by the type of reason the leave was taken or needed. For example, Question A5 asks the reason why the respondent took leave, and Question B6 asks an analogous question for leave needing. The vast majority[[6]](#footnote-7) of responses fell into one of six reasons:

1. Own Illness (variables have suffix \_own)
2. Maternity Disability (suffix is \_matdis)
3. Bonding with a newborn child (suffix is \_bond)
4. Caring for an ill child (suffix is \_illchild)
5. Caring for an ill parent (suffix is \_illparent)
6. Caring for an ill spouse (suffix is \_illspouse)

These are the categories used throughout the model to disaggregate by leave type.

***Reference periods of 12 months and 18 months.*** The survey often asks questions twice, once with a reference period of the past 18 months and once for the past 12 months (for example, A4 asks “For how many total reasons did you take leave in the last 18 months”. A4a asks “For how many total reasons did you take leave in the last year”). ACS and CPS have a reference period of 12 months. To maintain a consistent interpretation of model results, we only use information from leaves taken/needed within the last 12 months.

***Handling Multiple Leaves Taken/Needed.*** Questions A4 and B5 ask for the number of different reasons the respondent took/needed to take leave last year. If the respondent indicates multiple reasons, they may receive the succeeding questions twice in two separate loops. In the first loop of A5-A19 they are asked about the longest leave they took. Most pertinently for our model are the questions regarding the leave reason and length. If they responded to A20 indicating the most recent leave they took was a different reason than the longest leave, they then receive a second loop of A5-A19 regarding the most recent leave.

However, no questions about the third or further reasons are asked.[[7]](#footnote-8) To correct for this, our model performs an “intra-FMLA” imputation of these censored leave types and lengths. The details of this procedure are discussed later in Section X.

Section B of the survey is structured differently for leave needers. Respondents to this section were given up to three loops of questions B6-B20. In the first loop, they are asked about the most recent reason they needed leave. If they indicated multiple reasons for needing leave in B5, they receive a second/third loop for the other reasons. The length of leave they would have needed is not asked, although this created minimal data loss.[[8]](#footnote-9) While the model does perform intra-FMLA imputation on types of leave needed, it changes little of the underlying data.

It is also possible that for each reason a respondent took/needed leave, they took/needed the leave on multiple separate occasions. Each occasion’s length is not separately asked about. Rather, A19 reports the total amount of time taken off in the reference period for that reason. This length is combined across all occasions. Leave length is a pivotal variable to precisely estimate in our model, and so we do not disaggregate by separate leave occasions as a result. The model’s estimated leave lengths are interpreted as the total length of leave taken across all occasions as a result.

There is ambiguity in how a respondent interprets questions regarding multiple leaves, which poses a validity concern to the model’s estimation of number of leaves taken. Respondents are not strictly confined to the six categories of reasons mentioned above for number of reasons A4 and B5. As a result, there is a strong possibility that the respondent’s interpretation of “reasons” will often be incongruous. Multiple leave taking is an important part of accurately estimating a population’s leave taking behavior. This unobserved incongruous interpretation threatens the use of these questions as a valid measure of multiple leave taking behavior as our model defines it. In addition, the distinction between separate leave occasions and separate leave reasons is potentially difficult for respondents to make. This distinction is not reinforced at A4 and B5, and is another possible source of bias. Finally, references to “most recent” and “longest” leaves leave ambiguous whether the reference is to the most recent/longest *occasion* or the *reason*.

**3.X.2 ACS Background**

**3.X.3 CPS Background**

## 3.2 Main Simulation Function

The main function to run a simulation is policy\_simulation(), which is defined within the 0\_master\_execution\_function.R file. It accepts X arguments. All arguments are optional except for the filenames of the four input datasets in csv format. Exhibit 3 presents a high-level summary of function arguments. More detail on argument expected values, format, and interpretation are given the in the Parameter Codebook accompanying this model. The function returns the ACS dataset modified with additional columns representing simulated leave-taking behavior and program participation for each individual, and summary analysis of the resulting simulation.

Exhibit 7: Summary of Input Arguments to policy\_simulation() Function

| Argument Name | Default | Description |
| --- | --- | --- |
| fmla\_csv | - | Cleaned FMLA data in csv format |
| acs\_house\_csv | - | Household Level ACS data in csv format |
| acs\_person\_csv | - | Individual Level ACS data in csv format |
| cps\_csv | - | CPS data in csv format |
| impute\_method | KNN1 | Method to use for FMLA to ACS imputation. |
| xvars | [Long list of 19 variables] | xvars for imputation method to use. |
| leaveprogram | FALSE | Presence or absence of leave program |
| base\_bene\_level | 1 | proportion of pay received as part of program participation |
| bene\_effect | FALSE | Whether to apply simulation of behavioral cost to applying to state program |
| topoff\_rate | 0 | proportion of employers engaging in top-off substitution of paid leave with program benefits |
| topoff\_min\_length | 0 | Min length of leave required for top-off behavior |
| dependent\_allow | 0 | weekly dependent allowance for those with children |
| full\_particip\_needer | FALSE | whether or not leave needers always take up benefits |
| own\_uptake | 1 | user-supplied benefit uptake rate for a given type of leave. |
| illspouse\_uptake | 1 | user-supplied benefit uptake rate for a given type of leave. |
| illchild\_uptake | 1 | user-supplied benefit uptake rate for a given type of leave. |
| illparent\_uptake | 1 | user-supplied benefit uptake rate for a given type of leave. |
| matdis\_uptake | 1 | user-supplied benefit uptake rate for a given type of leave. |
| bond\_uptake | 1 | user-supplied benefit uptake rate for a given type of leave. |
| waiting\_period | 0 | how long in working days must leave takers wait to claim leave benefits |
| clone\_factor | 0 | Create clones of ACS records |
| ext\_base\_effect | TRUE | Whether to apply base leave extension behavior in presence of program. standard leave extension effect from ACM model |
| extend\_prob | 0 | additional leave extension effect: probability of extension |
| extend\_days | 0 | additional leave extension effect: fixed days of extension |
| extend\_prop | 1 | additional leave extension effect: proportionate extension |
| maxlen\_own | 60 | max number of leave type days benefits can be claimed in a year |
| maxlen\_illspouse | 60 | max number of leave type days benefits can be claimed in a year |
| maxlen\_illchild | 60 | max number of leave type days benefits can be claimed in a year |
| maxlen\_illparent | 60 | max number of leave type days benefits can be claimed in a year |
| maxlen\_matdis | 60 | max number of leave type days benefits can be claimed in a year |
| maxlen\_bond | 60 | max number of leave type days benefits can be claimed in a year |
| maxlen\_DI | 240 | max number of bond, ill relative leave days benefits can be claimed in a year |
| maxlen\_PFL | 120 | max number of matdis, own leave days benefits can be claimed in a year |
| maxlen\_total | 360 | max number of total days benefits can be claimed in a year |
| week\_bene\_cap | 1000000 | max weekly benefits that can be collected |
| week\_bene\_cap\_prop | NULL | option to cap max weekly benefits that can be collected at a proportion of the mean weekly wage |
| week\_bene\_min | 0 | min weekly benefits that can be collected |
| fmla\_protect | TRUE | Indicates whether or not leaves that are extended in the presence of a program that originally were less than 12 weeks in length are constrained to be no longer than 12 weeks in the presence of the program |
| earnings | NULL | Minimum earnings in dollars in past 12 months |
| weeks | NULL | Minimum weeks worked in past 12 months |
| ann\_hours | NULL | Minimum total number of hours worked in past 12 months |
| minsize | NULL | Minimum number of employees working at their employer |
| weightfactor | 1 | Multiply ACS weights by a certain number |
| random\_seed | NULL | set random seed if user wishes analyses to be replicable |
| SELFEMP | FALSE | Whether to include self employed workers in ACS data set |
| FEDGOV | FALSE | Whether to include federal gov't workers in ACS data set |
| STATEGOV | FALSE | Whether to include state gov't workers in ACS data set |
| LOCALGOV | FALSE | Whether to include local gov't workers in ACS data set |
| formula\_prop\_cuts | NULL | Specification for formulaic benefits based on state mean wage |
| formula\_value\_cuts | NULL | Specification for formulaic benefits based on absolute wage values |
| formula\_bene\_levels | NULL | Proportion of pay those under each cut receive |
| elig\_rule\_logic | NULL | Description of the logic used when multiple eligibility criteria are specified. |
| own\_elig\_adj | 1 | user-supplied eligibility adjustment for a given type of leave. |
| illspouse\_elig\_adj | 1 | user-supplied eligibility adjustment for a given type of leave. |
| illchild\_elig\_adj | 1 | user-supplied eligibility adjustment for a given type of leave. |
| illparent\_elig\_adj | 1 | user-supplied eligibility adjustment for a given type of leave. |
| matdis\_elig\_adj | 1 | user-supplied eligibility adjustment for a given type of leave. |
| bond\_elig\_adj | 1 | user-supplied eligibility adjustment for a given type of leave. |
| kval | 3 | KNN\_multi method only. Number of neighbors to take for imputation |
| makelog | TRUE | whether to produce log file of execution or not |
| ext\_resp\_len | FALSE | test parameter for alt length rand draw. To elaborate on |
| len\_method | TRUE | test parameter for alt length rand draw. To elaborate on |
| saveCSV | TRUE | save CSV file of output data set |
| useCSV | TRUE | use CSV versions of input files and conduct cleaning |
| saveDF | FALSE | save RDS file of output data set |

## 3.3 Cleaning Functions

This section describes the functions used to prepare the three raw datasets (ACS, FMLA, and CPS) for use in the model. The function first reads in the raw datasets and cleans them using the clean\_fmla(), clean\_acs()and clean\_cps() functions which are all defined in the 1\_cleaning\_functions.R file. All together, these functions result in cleaned datasets formatted to be used conveniently by the functions in the rest of the model.

### 3.3.1 The clean\_fmla()Function

The clean\_fmla()function loads the raw FMLA survey data file, and derives a large number of variables for later use in the simulation engine. It then outputs a clean FMLA data set. Full detail on the derived variables is available in the accompanying Derived Data Codebook. Here, we summarize what variables we have derived, and the important details to know about their construction.

***FMLA eligibility status.*** Eligibility for FMLA protection (coveligd) is used as a control variable throughout the original ACM model. In our model, it is an available control variable for use in the main FMLA to ACS imputation function. Also, when conducting the intra-FMLA imputation of unobserved leave types taken (see [section number]), we follow ACM’s logit model for selecting type of leave, which includes FMLA eligibility as a control variable.

***Demographic and employment variables.*** We clean and standardize a wide variety of demographic variables that are also present in the ACS. These include race, age, gender, number of children, education, and more. This is to ensure the variables follow the exact same format in both data sets. For the same reason, we also standardize characteristics of employment (such as weekly hours worked, union membership, and public/private sector employment status).

***Benefits/pay received while on leave.*** The proportion of pay received variable (prop\_pay) is a numeric variable derived from the categorical responses to A50. Some of the A50 responses represent a range of proportions (such as “More than half but less than three-quarters”). We assign the mid-point of the range for these values.

***Financial sensitivity for leave taking.*** The demand elasticity of taking leave varies from individual to individual. Individuals with sufficient income to take all leave they required will likely not increase leave taking. A number of questions ask in various ways about whether more pay would have resulted in an individual taking more or all of their needed leave. We build a composite binary variable (resp\_len) from these questions to indicate whether a respondent’s leave taking behavior is financially sensitive.[[9]](#footnote-10) The original ACM model used a narrower definition only incorporating question B15. However, leave takers could also experience financial sensitivity with the presence of a leave program. We used questions A23c, A53g, A55, and A62a for leave takers. We preserve ACM’s original definition in our unaffordable variable. The simulated counterfactual leave taking behavior of individuals (goal 2 of the model described above) is highly dependent on this variable.

***Presence or absence of a public leave program for respondents.*** The behavioral impact of a leave program is an important part of our model. The introduction of a paid leave program offers financial incentive for leave taking. A prerequisite for simulating this effect is observing the presence or absence of this program in the FMLA data set. The FMLA survey was administered in 2011, during which some states did offer paid family leave or temporary disability insurance for leave takers. As a result, the observed leave lengths are mixed; we observe leave length of some respondents in the presence of the program, and others in the absence of one.

Fortunately, FMLA data does indicate whether individuals benefited from a state program or not. Questions A48b and A48c ask whether any of the pay a respondent received was from a state paid family leave or temporary disability program. We use these responses to indicate the presence or absence of a paid leave program for each FMLA observation. We refer to the absence of program scenario as the “status quo” scenario and the presence of program as the “counter-factual” scenario.

***Reason leave is taken/needed.*** The FMLA asks respondents the number of different reasons leave was taken/needed, what their reasons were, and what their total length was. For each of the six different leave types, we define four binary variables:

1. take\_[\*type\*] – Most recent leave taken is of this type
2. need\_[\*type\*] - Most recent leave needed is of this type
3. long\_[\*type\*] – Longest leave taken is of this type
4. type\_[\*type\*] – Most recent leave taken or most recent leave needed is of this type

For each leave type, we also define two numeric integer variables:

1. length\_[\*type\*] – Total length of most recent leave
2. longlength\_[type] - Total length of longest leave

Maternal disability leave and child bonding leaves share the same code for A5. We distinguish between these two by incorporating A11; whether the mother required doctor’s care during the leave. If they did require doctor’s care, it was considered a maternity disability leave. If not, it was considered a child bonding leave. All other leave types were uniquely identifiable by A5 answer codes.

### 3.3.2 The clean\_acs()and clean\_cps()Functions

The clean\_acs()function takes the raw ACS PUMS 2012-2016 5 year data retrieved from Census.[[10]](#footnote-11) The data set comes in two files; a household-level file and a person-level file. We begin by merging selected demographic variables from the household-level file into the person-level file. The remainder of the model uses solely this modified person-level file. We then create variables identical in definition to the corresponding demographic/employment variables available in the FMLA survey. We then filter the ACS data set to only contain civilian employed individuals over the age of 18. It then outputs the clean ACS data set.

Similarly, the clean\_cps()function takes the raw CPS ASEC 2014 supplement file, and cleans a number of demographic variables to match the definitions of corresponding variables in the ACS data set. It then outputs the clean CPS data set.

### 3.3.3 The impute\_cps\_to\_acs()Function

### Some variables contained in the FMLA data are not available in the ACS. The impute\_cps\_to\_acs()function takes the cleaned CPS and ACS data sets as inputs, and uses logit and ordinal logit regressions to probabilistically impute the required variables from CPS to ACS. Further details on how logit and ordinal imputations are performed are discussed with in section 3.X with the imputation functions. It then outputs an ACS data set with the imputed variables added on. These imputations use various sets of demographic variables available in both surveys. The four imputed variables are the following:

***Whether an individual is paid hourly or not (paid\_hrly).***  This is a dummy variable indicating whether or not an individual is paid by the hour. It is imputed with a logit regression.

***The number of employers an individual has worked for in the past 12 months (num\_emp).*** This is an integer variable taking the values 1, 2, or 3. This is imputed via an ordered logit regression to ACS.

***Number of weeks worked in the past 12 months (weeks\_worked).*** This is an integer variable from 0-52 that represents the number of weeks worked in the past 12 months. This element is found in the ACS data set, but is a categorical variable only representing ranges of weeks. The ACS indicates the number of annual weeks worked by an individual within bins of <13 weeks, 14-26 weeks, 27-39 weeks, 40-47 weeks, 48-49 weeks and 50-52 weeks. It is available in CPS with greater specificity, the integer value of weeks worked. We do an ordinal imputation of the integer value from the CPS, conditional on the ACS observation’s respective range of weeks worked.

***Firm size of primary employer (emp\_size).*** This is a categorical value representing ranges of employer sizes, in terms of employees employed. The size range is imputed via an ordered logit regression to ACS. Then, a value within their assigned range is randomly chosen to each ACS individual.

## 3.4 Pre-Imputation

This section describes the functions defined in the 2\_pre\_impute\_functions.R file to perform pre-imputation manipulations of the ACS and FMLA data sets used in the model. The impute\_intra\_fmla()function and acs\_filtering() function are defined in this file. Together, these functions result in FMLA and ACS datasets ready for FMLA to ACS imputation.

### 3.4.1 The impute\_intra\_fmla()Function

This function is motivated by the importance of estimating the type and length of leaves. Within the FMLA, leave type and length is only available for a maximum of two unique leaves – their last leave taken and their longest leave taken – in the previous 12 months for an individual survey respondent. Therefore, there are some circumstances under which we do not have complete leave length or type information for a respondent. To understand when this might occur, consider the examples presented in Exhibit 4. All leave information is available for individuals A and B, who took just one and two leaves, respectively, in the previous 12 months. On the other hand, although individual C also took two leaves, her latest leave taken also happened to be her longest leave over the previous 12 months. Therefore, we do not have information regarding her other (shorter) leave that she took. In the case of individual D, although we have information for two of her leaves, we do not know anything about her third leave.

Exhibit 9: Example Leave Information Responses in FMLA

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Individual ID | Number of Leaves | Latest Leave Length | Latest Leave Type | Longest Leave Length | Longest Leave Type |
| A | 1 | 5 | Own Health | 5 | Own Health |
| B | 2 | 6 | Ill Spouse | 12 | Ill Child |
| C | 2 | 30 | Maternity | 30 | Maternity |
| D | 3 | 1 | Own Health | 8 | Ill Parent |

For individuals that indicated taking multiple leave, we need to infer the type of leave for each of those. This function accounts for this by using a probabilistic logit regression to impute the leave types. This is done using logistic regression with the specifications (dependent variables, sample conditionals and weights) identical to those used to perform the same task in the original ACM model. Using the estimates from these regressions, we apply them to each individual’s characteristics to generate a probability that they took each unobserved leave type. We then select randomly from these leave types in proportion to the logit-calculated probability until we have made up the difference between reported and observed number of leaves. Each leave is assigned a normalized probability proportional to the probability score estimated by the logit model above. Exhibit X below illustrates an example of this process for individual D from above.

Exhibit 9: Leave Imputation Example

*Step 0. Data observed in FMLA*

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Individual ID | Num. of Leaves | Take Own Health | Take Mat. Disability | Take Ill Spouse | Take Ill Child | Take Ill Parent | Take Child Bond |
| D | 3 | Y | N | N | N | Y | N |

*Step 1. Calculate probabilities of remaining leave types with logit regression*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Individual ID | P(Take Mat. Dis.) | P(Take Ill Spouse) | P(Take Ill Child) | P(Take Child Bond) |
| D | .70 | .20 | .40 | .10 |

*Step 2. Normalize probabilities to sum to 1.*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Individual ID | P(Take Mat. Dis.) | P(Take Ill Spouse) | P(Take Ill Child) | P(Take Child Bond) |
| D | .50 | .14 | .29 | .07 |

*Step 3. Create intervals based on normalized probabilities, generate random number*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Individual ID | Take Mat. Dis. | Ill Spouse | Take Ill Child | Take Child Bond | Rand # [0, 1] |
| D | [0, .50] | [.50, .64] | [.64, .93] | [.93, 1] | .60 |

*Step 4. Select leave type based on random number, and update FMLA data*

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Individual ID | Num. of Leaves | Take Own Health | Take Mat. Disability | Take Ill Spouse | Take Ill Child | Take Ill Parent | Take Child Bond |
| D | 3 | Y | N | **N Y** | N | Y | N |

*Step 5. Repeat steps 1-4 until all observations have as many observed leave types taken as reported.*

### 3.4.2 The acs\_filtering()Function

This is a short function that processes user input parameters to select a particular state from the ACS data set if not already filtered, and whether to consider “state of residence” or “state of work”.

The FMLA and the ACS are the key data sources driving the microsimulation model. However, each dataset is missing crucial information required for the model simulation. It is therefore necessary to “fill in the gaps” using imputation methods in order for the datasets to be useful. The next section explains why the imputations are needed and how they are performed in each dataset.

## 3.5 FMLA to ACS Imputation

This section describes the functions defined in the 3\_impute\_functions.R file, which perform the imputation of a set of leave variables from the FMLA to the ACS data set.

### 3.5.1 The impute\_fmla\_to\_acs()Function

The impute\_fmla\_to\_acs() function executes the actual imputation. The remaining functions specify different methods that can be used to perform the imputation. The output of this function is an ACS data set with columns representing leave taking/needing, and other variables exclusively available in FMLA that are required for the rest of the model.

To perform the FMLA to ACS imputation, the model has 7 different methods available: single nearest-neighbor, multiple nearest-neighbor, logistic regression, naïve Bayes classification, ridge classification, random forest classification, and support vector machine classification. Each have their own function, correspondingly: KNN1\_scratch(), KNN\_multi(), logit\_leave\_method(), Naive\_Bayes(), ridge\_class(), random\_forest(), and svm\_impute().

All methods require similar inputs and produce output with identical structure. Each function always requires a training data set, a testing data set, a list of dependent/independent variable names, and data filtering conditions as inputs. All functions will output the input testing data set with the following imputed variables added as additional columns:

* The six take\_\* variables
* The six need\_\* variables
* prop\_pay
* resp\_len
* unaffordable

Some of the method-specific nuances are listed below:

* KNN1\_scratch() uses a Euclidean distance function to find the nearest neighbor. All independent variables are normalized to 1, effectively assigning equal weight to the distance of all independent variables. No external packages are used.
* KNN\_multi() has an additional input, kval. This represents the number of nearest neighbors to find for each individual. It uses the same distance calculation and normalization methods as KNN1() to find neighbors. It currently use the “voting” imputation method; the mode value of the neighbors is imputed to the individual. No external packages are used.
* logit\_leave\_method()uses a logit model for all variables except for prop\_pay, which requires an ordinal logit. Logit regression is performed with the svyglm() function from the survey package. Ordinal logit regression is performed with the polr() function from the MASS package.
* Naive\_Bayes() uses the naiveBayes() function from the e1071 package. We use the default settings with epsilon smoothing disabled.
* ridge\_class()normalizes all independent variables to 1. prop\_pay is handled differently due to it being a ordered categorical variable. A separate binary variable for each prop\_pay category is created. A separate ridge classifier model is created for each of these binary variables, and probabilities estimated. The resulting probabilities are normalized to 1, and categories are assigned intervals based on that. Finally, a category is selected for imputation based on what interval a random number with range [0,1] falls into. This function uses the lm.ridge() function from the MASS package.
* random\_forest()uses the randomForest() function from the randomForest package. prop\_pay is handled differently due to it being a ordered categorical variable. The model uses a penalized intercept.
* svm\_impute()

[more text on selection of method after we get farther with performance testing of the different methods.]

### 3.5.2 Non-User-Specified Imputation Methods

Logistic and ordinal logistic regression are one available option for FMLA to ACS imputation, but are also the functions used in other parts of the model to perform other kinds of imputations. The imputation procedures are currently hard-coded (they do not change based on user specifications) into the model and are based on the logistic regression methods used in the original ACM model. There are three different hard-coded methods: runLogitEstimate(), runOrdinalEstimate(), and runRandDraw(), which correspond to logit, ordinal logit, and random draw imputation methods, respectively. Logit is used to impute binary dependent variables, while ordinal logit is used to impute categorical variables with 3 or more categories. Random draw is used to impute variables where the relevant FMLA subgroup lacks sufficient sample size to construct a meaningful regression model.

The runLogitEstimate()function takes a training data set, a testing data set, a list of dependent/independent variable names, and data filtering conditions as inputs. For each dependent variable, the function constructs logit coefficients based on the specified dependent/independent variables on the filtered training dataset and returns the set of parameter estimates. Using the coefficients from above, the function computes the implied probability values for each row in the filtered testing data set based on the same independent variables. It then simulates the dependent variable by comparing a random uniform draw with the computed probability and adds this result as a column to the data frame. Logit regression is performed with the svyglm() function from the survey package.

The runOrdinalEstimate()function takes a similar set of inputs, and uses the training data set to estimate coefficients for an ordinal logit in a similar fashion. Using these coefficients from above, the function computes the implied “cutoff” values for each of the categories. It then simulates the dependent variable by comparing a random uniform draw with the intervals computed above and adds this result as a column to the data frame. Ordinal logit regression is performed with the polr() function from the MASS package.

The runRandDraw()function takes a training data set, a testing data set, and data filtering conditions as inputs as well, but only a list of dependent variables as inputs. For each variable, randomly selects values from the filtered training data set to give to each observation in the filtered testing data set.

Hard-coded imputation is used at the following points of the model:

1. To impute relevant variables from CPS to ACS (logit/ordinal logit)
2. To perform an intra-FMLA imputation of leave taking within FMLA (logit).
3. To impute leave lengths for those lacking leave length (random draw).
4. To perform a counterfactual leave extension simulation (logit).

## 3.6 Post-Imputation Functions

This section describes the functions defined in the 4\_post\_impute\_functions.R file. The functions take the post-FMLA imputed ACS data set and perform a variety of manipulations and changes to it based on various user input parameters. Together, these functions produce a completed ACS data set with all simulations and modifications made.

There are 12 main functions in this file, performing one of three different kinds of modifications to the ACS data set: 1) simulating alternate leave taking behavior in the counterfactual scenario of an active leave program, 2) simulating program participation based on eligibility rules, benefit levels, and uptake parameters, or 3) calculating program benefits for those who elect to participate in the program. Some of these functions are intended to replicate a command from the original ACM model; the name of the function matches the original ACM model’s command name in those instances. All of these functions use the ACS data set as an input, make modifications to it, and output a modified ACS data set.

### 3.6.1 The LEAVEPROGRAM() Function

A key feature of the microsimulation model is to predict changes in leave behavior resulting from changes in leave program parameters such as the wage replacement rate, the number of days and the type of eligible leave etc. This counterfactual simulation is performed by LEAVEPROGRAM(). This function takes the post-imputation ACS data set and adjusts the leave taking behavior of those with financial sensitivity. It then returns the modified ACS data set. The function adjusts the “take\_x” variable for leave type x to 1 from 0 if the ACS individual worker needs leave type x and is deemed to be sensitive to financial incentives for leave taking. This is needed to implement the ACM model’s assumption that those who needed but did not take leave for financial reasons in absence of a state paid leave program will elect to take leave in the presence of such a program.

### 3.6.2 The impute\_leave\_length() Function

The impute\_leave\_length() function adds in leave lengths to ACS data through a random draw from applicable FMLA leave takers. It follows the ACM approach, which proceeded as follows. The ACM model assumes that that length of leave taken is related to both the type of leave and whether a state leave program was available. A cross-tabulation of the mean leave lengths by these characteristics in FMLA data supports this assumption (see the large variations observed in Exhibit X below). The ACM model did not directly impute leave length via regression like other leave variables were. Instead, the ACM model conducted a random draw from the lengths of the same type of leaves observed in FMLA. ACM wished to also draw randomly conditional on receiving state pay or not. However, only Own Health leaves had sufficient sample size in the “received state pay” subgroup (see Exhibit X below). We follow ACM’s random draw method and subgroup strategy to impute leave length in this function.

**Exhibit X. Mean Leave Length in Days, by Leave Type**

|  |  |  |  |
| --- | --- | --- | --- |
| Leave Type | Received State Pay | Did not receive state pay | Overall |
| Own Health | 58.7 | 34.2 | 36.0 |
| Ill Spouse | 9.5 | 18.9 | 18.5 |
| Ill Child | 8.9 | 12.3 | 11.7 |
| Ill Parent | 13.4 | 20.1 | 19.5 |
| Maternal Disability | 73.5 | 65.9 | 65.1 |
| Child Bonding | 25.3 | 29.9 | 30.2 |

Source: FMLA 2012 Survey

**Exhibit X. Number of FMLA Observations Reporting Leave Length**

|  |  |  |
| --- | --- | --- |
| Leave Type | Received State Pay | Did Not Receive State Pay |
| Own Health | 69 | 620 |
| Ill Spouse | 4 | 93 |
| Ill Child | 6 | 59 |
| Ill Parent | 6 | 111 |
| Maternal Disability | 6 | 90 |
| Child Bonding | 7 | 91 |

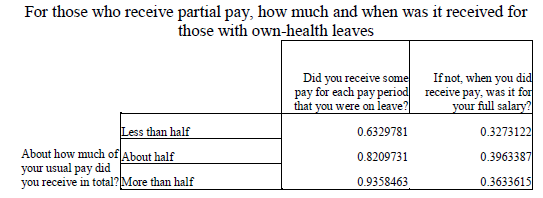
### 3.6.3 The CLONEFACTOR() Function

This function allows a user to reduce variance due to simulation-induced randomness (such as the probabilistic random draws done with logit estimations) by artificially “cloning” observations. This is done by randomly selecting with replacement a user-specified proportion of the ACS data set, duplicating those observations, and appending the duplicated observations to the original ACS data. For state-level estimates, the simulation-induced variance is very low from our own tests, and ACS sample size is generally sufficient. We don’t anticipate users needing to employ this option often.

### 3.6.4 The PAY\_SCHEDULE() Function

Employer pay received is important to establish before simulating individuals’ decision to participate in the leave program. But the schedule of pay received over the duration of the leave is not always constant in reality. Some employers will only cover part of the leave. PAY\_SCHEDULE()is a replication of the ACM model’s simulation method for assigning the schedule for which an individual receives employer pay for their leave. The function assigns individuals to receive one of three different pay schedules: 1) received pay for the entirety of the leave, 2) received full pay for part of the leave, or 3) received partial some pay for some part of the leave. These are assigned with probabilities derived from the proportionate responses to two questions from a 2001 Westat survey: “Did you receive some pay for each pay period that you were on leave?” and “If not, when you did receive pay, was it for your full salary?” Schedule 1 corresponded to those who responded “Yes” to the first question. Schedule 2 corresponded to those who responded “No” to the first question and “Yes” to the second. Schedule 3 corresponded to those who responded “No” to both questions. These schedules’ probabilities are calculated conditional on how much pay individuals reported receiving overall (prop\_pay). The table of conditional probabilities observed in the Westat survey is shown in Exhibit X below.

### Exhibit X. Pay Schedule and Amount in the Westat 2001 Survey



Source: ACM Model Documentation

The purpose for assigning these schedules is to establish when in their leave an individual might fully exhaust their employer benefits, and then become eligible to apply for state paid leave benefits. The prop\_pay variable establishes the *total* proportion of wages the employer will give them over the leave period, but is uninformative as to *how* that pay is distributed over the course of the leave.

For example, consider an individual who normally earns a daily wage of $100 and has a prop\_pay value of .2 (employer pays them 20% of their normal wage for that leave overall). They wish to take leave for 10 days. So we know they will receive $200 in total, but do they receive $20 each day for all 10 days (Schedule 1), do they receive $100 the each day until they hit $200 (Schedule 2), or something in between (Schedule 3)? This is relevant to determine because as soon as an individual exhausts employer benefits (in other words, starts receiving $0 per day from their employer), they are assumed to become eligible to receive state benefits. An illustrative example of how employer pay is distributed for each schedule is given in Exhibit X below.

**Exhibit X. Example of Pay Schedules**

|  |  |  |  |
| --- | --- | --- | --- |
| All individuals take 10 days of leave, are normally paid a daily wage of $100, and receive 20% of their pay in total when on leave. This table shows how each of the pay schedules distributes the $200 they receive from their employer while on leave. | | | |
| **Day of Leave** | **Individual A**  **(Gets Pay Schedule 1)** | **Individual B**  **(Gets Pay Schedule 2)** | **Individual C**  **(Gets Pay Schedule 3)** |
| Day 1 | $20 | $100 | $50 |
| Day 2 | $20 | $100 | $50 |
| Day 3 | $20 | $0 | $50 |
| Day 4 | $20 | $0 | $50 |
| Day 5 | $20 | $0 | $0 |
| Day 6 | $20 | $0 | $0 |
| Day 7 | $20 | $0 | $0 |
| Day 8 | $20 | $0 | $0 |
| Day 9 | $20 | $0 | $0 |
| Day 10 | $20 | $0 | $0 |
| **Day benefits exhaust** | **Never** | **Day 3** | **Day 5** |

Schedule 1 individuals never exhaust benefits. Schedule 2 individuals exhaust benefits after days, rounded to the nearest whole day. Schedule 3 individuals exhaust benefits after days, also rounded.

### 3.6.5 The ELIGIBILITYRULES() Function

The ELIGIBILITYRULES() function establishes which individuals in the ACS are eligible for the simulated leave program, and begins the process of determining who will participate in it. The user can define eligibility thresholds for four different criteria: minimum earnings in the last 12 months (the earnings parameter), minimum weeks worked in the last 12 months (weeks), minimum hours worked in the last 12 months (ann\_hours), and minimum firm size of employer (minsize) . The user can also specify the logic all combinations of these eligibility criteria with “and” or “or” operators (i.e. individuals must meet minimums for ann\_hours AND weeks OR minsize). This is done using the elig\_rule\_logic parameter. The function generates a binary variable, eligworker, to note each individual’s eligibility for the program.

The function next calculates the proportion of normal wages each individual will receive as benefits. Under the default simulation settings, this is a flat rate for all participants specified by the base\_bene\_level parameter. However, there is also the option to implement a formulaic distribution of benefits by income brackets. This is handled by the FORMULA() subfunction, which is a replication of an original ACM command.

The FORMULA() subfunction takes two parameter inputs from the user; formula\_bene\_levels and either formula\_prop\_cuts or formula\_value\_cuts. Passing a list of numbers with formula\_value\_cuts means the list will be interpreted as the fixed dollar values that divide up income brackets. Alternatively, passing the list with formula\_prop\_cuts means the list will be used to create income brackets based off of proportionate income relative to the population’s mean income. For either, the lists passed must be made up of positive numbers in ascending order.

The formula\_bene\_levels parameter represents the proportion of normal wages each income bracket will receive while collecting benefits from the leave program. This list should be one number longer than the formula\_prop\_cuts or formula\_value\_cuts list. Based on these parameters, the subfunction modifies the prop\_pay value of each individual based on which bracket they fall into. We illustrate two examples of how this subfunction works to determine proportion of pay received in Exhibit X below.

**Exhibit X. Example of Formulaic Benefits Calculation**

*Example 1: Brackets based on fixed income*

Formula to implement:

|  |  |  |  |
| --- | --- | --- | --- |
| Annual Income | $0 - $20,000 | $20,001 - $60,000 | >$60,000 |
| **Proportion of pay received** | .6 | .5 | .4 |

User input parameters:

* formula\_value\_cuts:(20000, 60000)
* formula\_bene\_levels:(.6, .5, .4)

*Example 2: Brackets based on proportionate income relative to population mean income*

Formula to implement:

|  |  |  |  |
| --- | --- | --- | --- |
| Income as proportion of pop. mean income | 0 - .5 | .5 - .75 | >.75 |
| **Proportion of pay received** | .6 | .5 | .4 |

User input parameters:

* formula\_value\_cuts:(.5, .75)
* formula\_bene\_levels:(.6, .5, .4)

After calculating possible benefits, the function creates a participation indicator variable (particip). This variable will be modified by later functions, but the variable’s starting value is set by the following logic: eligible workers who would earn more benefits than they would receive from their employers, or those workers who will exhaust employer benefits before the end of their leave will participate in the program.

### 3.6.6 The EXTENDLEAVES() Function

Like the ACM model, our model assumes that the length of leave taken will be longer in the counterfactual presence of a leave program. To account for this behavioral effect, the EXTENDLEAVES() function simulates the extension of leaves with multiple different methods, which can be applied simultaneously if desired. These methods are based upon the ACM model’s leave extension commands.

The ACM model had two different leave extension commands: a "base" leave length extension effect, and an extension based on a linear model with user-specified values. The binary ext\_base\_effect parameter is used to enable/disable this ACM model extension effect in our own model. The ACM base extension effect works as follows:

1. ACM uses a logistic regression to impute from FMLA to ACS a binary variable (called longerLeave in our model) indicating whether an individual takes a longer leave in the presence of a counterfactual leave program. ACM runs a logit regression in FMLA data on question A55 (“Would you take a longer leave if you received some/additional pay?”) with some demographic characteristics as left-hand-side variables. ACM then calculates the probability of longer leave taking by applying the resulting coefficients to ACS observations, and probabilistically determines the longerLeave based on these probabilities.
2. For those ACS individuals for whom a “yes” was imputed to longerLeave, ACM applies the following behavioral effects:
   1. For workers with leave lengths shorter than the waiting period of the leave program, the leave is extended by 1 week. We alter this to (program waiting period + 1 week) in our adaptation.
   2. For workers who do not receive any employer pay or who exhaust their employer pay and then go on the program: The probability of extending a leave using program benefits is set to 25 percent; and for those who do extend their leave, the extension is equal to 25 percent of their length in the absences of a program. This is unchanged in our adaptation
   3. For workers who exhaust program benefits and then receive employer pay. In this case the simulator assigns a 50 percent probability of taking an extended leave until their employer pay is exhausted. This effect is omitted from our adaption as this scenario cannot occur; our model makes it impossible for someone to first collect program benefits, then employer pay.

In addition to a base leave extension effect, ACM allows the user to extend leaves based on a linear model with three user parameters: Extend each leave by additional days with probability, where x is the original leave length. The user can enable this alternative leave extension effect in our model with the corresponding user parameters: -> extend\_days, -> extend\_prop, -> extend\_prob.

Finally, like the ACM model, we also provide users with the ability to apply a behavioral constraint based on FMLA protection with the fmla\_protect parameter. If enabled, leaves that are extended in the presence of a program that originally were less than 12 weeks in length are constrained to be no longer than 12 weeks in the presence of the program. Since the FMLA statute extends only to 12 weeks of leave, this effect assumes that those taking leaves close to that limit originally will not extend leaves beyond the protection of FMLA for the security of their jobs.

In addition, we offer two methods of our own that users can use to extend leave lengths.

**Resp\_len with random draw** – We conduct two random draws, one draw for the counterfactual length and one for the status quo length. The status quo draw is from FMLA subsample conditional on resp\_len = 0. The counterfactual length for those in ACS with resp\_len = 0 subsample is same as status quo. The counterfactual length for those in ACS with resp\_len = 1 is drawn from the FMLA subsample conditional on resp\_len = 1, and the drawn length being greater than counterfactual draw.

**Resp\_len with mean average added** – We conduct one random draw for status quo from FMLA subsample conditional on resp\_len = 0 for ACS observations. We find the proportional difference in means between resp\_len = 1 and resp\_len = 0 subgroups in the FMLA, then multiply the status quo draw by that difference for ACS observations.

### 3.6.7 The UPTAKE() Function

At this point in the simulation, we’ve established what the leave lengths will be, and what benefits individuals could receive from their employer and the simulated paid leave program. However, not all eligible individuals will elect to participate in the program for economic or behavioral reasons. The participation indicator variable (particip) was previously created in the ELIBIGLITYRULES() function. The UPTAKE() function will modify this variable further and derive additional variables to track participation.

This function introduces seven variables to track the number of days an individual participates in the program. particip\_length is the total number of days that an individual collected leave program benefits across all types of leaves. The six plen\_\* variables measure the number of days benefits are collected for a single type of leave. The wait\_period parameter represents the days of leave individuals must wait before beginning to collect program benefits. If the leave is longer than the wait period (i.e. length\_\* > wait\_period), the plen\_\* variable will be length\_\* - wait\_period. If it’s shorter or equal to the wait period, plen\_\* will be 0. Individuals must also have particip = 1 to have any non-zero values for these seven variables.

Participation is also modified by the six \*\_uptake parameters. Each of these is a number 0-1 that represents the proportion of eligible and economically incentivized leave takers that will in fact enroll in the program. Economically incentivized refers to when they receive more program benefits than their employer pays in leave.

The full\_particip\_needer parameter modifies leave needer take-up behavior. Recall that due to the LEAVEPROGRAM() function, leave needers due to affordability reasons now take leave in the presence of the program. If enabled, this parameter will always have leave needers always take up benefits (rather than with probability specified by the appropriate \*\_uptake parameter)

when economically incentivized. Then, the check\_caps() subfunction ensures that length of participation in the program conforms with program participation length limits, and truncates participation if it exceeds program limits.

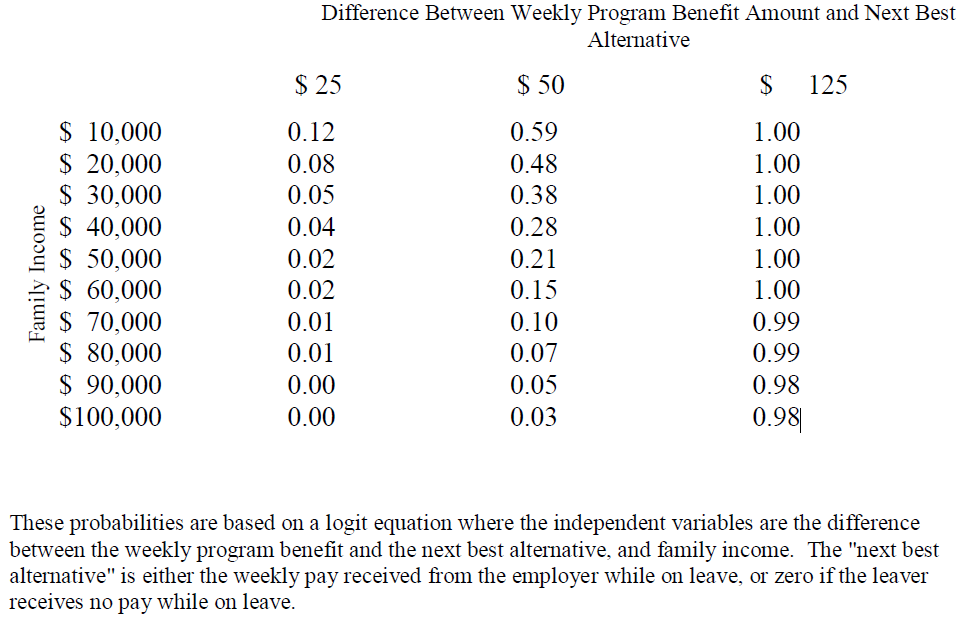
### 3.6.8 The BENEFITS() Function

This function calculates the base level program benefits and employer leave pay each individual would receive in ACS based on their wages and leave taking. The following functions modify this to account for various behavioral effects and program rules.

### 3.6.9 The BENEFITEFFECT() Function

This function accounts for some behavioral "cost" of applying for the leave program (time taken to apply to program, go through eligibility screening, etc.) when deciding between employer paid leave and program. Behavioral probabilities follow the ACM model methodology. The ACM model calculates uptake probability observed in the aforementioned 2001 Westat survey. These probabilities are conditional on difference in value of benefits to the program and employer pay, and family income. See Exhibit X below for ACM’s calculated probabilities.

Exhibit X. Probability of Participating for Selected Values of Benefit/Wage

Differential and Family Income

If this function is enabled, those ACS individuals currently participating in the program are assigned the probability value from the above table corresponding to that individuals benefit/wage differential and family income level. Participating individuals are then switched to non-participants with probability 1 minus the assigned probability value.

### 3.6.10 The TOPOFF() Function

This function simulates a possible firm-level behavioral effect of a program that the ACM model refers to as “top-off” behavior. “Topping off” is when employers who would pay their employees

100 percent of wages while on leave would instead require their employees to participate in the program. Employers would then "top-off" the program benefits by paying the difference between program benefits and full pay. User can specify percent of employers that engage in this, and minimum length of leave that employers would consider engaging in such behavior.

### 3.6.11 The DEPENDENTALLOWANCE() Function

This is a short function that adds a flat amount to participant’s weekly benefits for those participants with dependent children. The amount is specified by the user parameter dependent\_allow.

### 3.6.12 The DIFF\_ELIG() Function

Eligibility to this point in the model is indiscriminate to leave type; either an individual is eligible to receive benefits for all 6 leave types, or none of them. However, some state programs have differential eligibility by leave type. For example, New Jersey's private plan option allows employers to opt out of the state Disability Insurance program by offering a private disability insurance plan. About 30% of the employed population of NJ is covered by a private plan. This means about 70% of the eligible population can receive benefits for any kind of leave, while the remaining 30% can only receive benefits for the ill relative or child bonding types.

To accurately simulate this, the DIFF\_ELIG() function allows the user to adjust eligibility for different kinds of leaves. This is done by removing some specified proportion of participation for specific leave types at random from the population. The proportion for each leave type is specified by the six \*\_elig\_adj parameters.

### 3.6.13 The CLEANUP() Function

This function is the final step of the model which performs a final set of consistency checks and modifications. This function checks that participating leave lengths conform with program maximums, weekly benefit payments are capped at the program maximum, either at a fixed value or as a function of population mean weekly wage, and participating individuals receive at least the weekly minimum benefit. Then, this function calculates the benefits broken out by each leave type.

## 3.X Output Analysis Functions

This section describes the functions defined in the 5\_output\_analysis\_functions.R file.

## 3.6 GUI functionality

## 3.7 File Summary

This section presents a brief summary of what each file contains, which can often be helpful from a developer’s perspective for ease-of-reference. Exhibit X summarizes the information already presented earlier in this chapter but categorized according to the file in which each function is contained.

Exhibit 10: Summary of Model Files

|  |  |  |
| --- | --- | --- |
| File Name | Function | Description |
| 0\_master | impute\_intra\_fmla | This function Cleaned FMLA data in csv format |
| LEAVEPROGRAM |  |
| 1\_cleaning\_functions | - | Household Level ACS data in csv format |
| 2\_impute\_functions | - | Individual Level ACS data in csv format |
| 3\_post\_impute\_functions | - | CPS data in csv format |
| leaveprogram | True | Boolean indicating whether model computes counterfactual leave program |
| GOVERNMENT |  |  |
| SELFEMP |  |  |
| impute\_method |  |  |
| sample\_prop | NULL | Desired sample proportion to be used from ACS (to speed up runtime) |
| sample\_num | NULL | Desired sample size to be used from ACS (to speed up runtime) |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

## 3.X Suggested Readings

In addition to the documentation provided with this model, we suggest reviewing the following external documents:

* Original ACM model documentation
* ACS codebook
* CPS codebook
* FMLA codebook

# Chapter 4. Python Implementation

## 3.1 Overview of Code Structure

## 3.2 Main Simulation Function

## 3.3 Pre-processing Functions

## 3.4 Imputation Functions

### 3.4.1 CPS Imputation

### 3.4.2 FMLA Imputation

### 3.4.2 ACS Imputation

## 3.5 Simulation Functions

### 3.5.1 Logistic Regression

### 3.5.2 K-Nearest Neighbors

### 3.5.3 Random Forest

## 3.6 GUI functionality

# Chapter 5. Algorithms

This chapter describes the various algorithms available to the programmer at each stage of the model code.

## 5.1 Nearest Neighbor

Nearest Neighbor (NN) methods are based on the simple idea that “nearby” data points tend to be similar to each other. The methods work by finding the datapoints that are closest to a given observation in -dimensional space (where is the number of features or predictive variables used) and then tallying the classes of those neighbors. The data point is then predicted to belong to the class that appears the most among its neighbors.

The different variants of NN methods differ according to how they define the measure of “distance”, which is in turn used to define what points are “nearby”. Exhibit 5 presents a selection of popular distance metrics commonly used in NN methods. Note that each metric has its own tunable parameters, and some can be viewed as special cases of another. For example, the Minkowski distance is equal to the Manhattan distance if and the Euclidean distance if .

Exhibit 8: Alternative Distance Metrics for Nearest Neighbor Methods

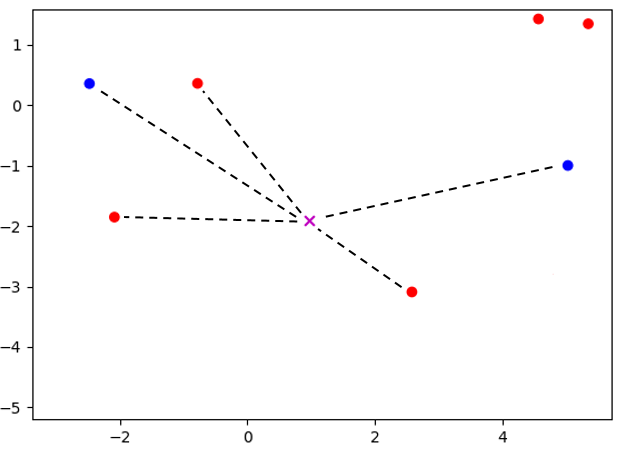
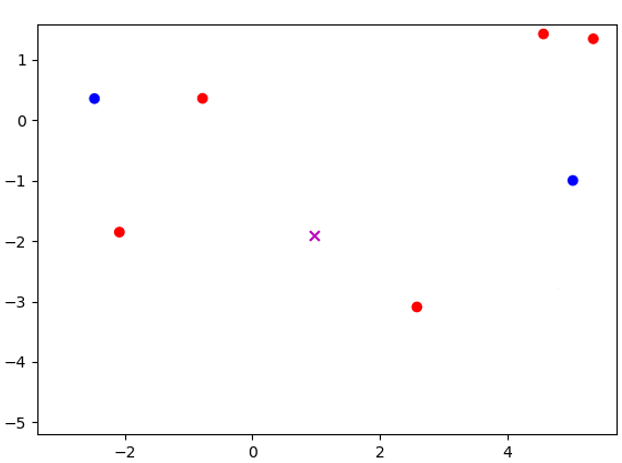
|  |  |
| --- | --- |
| Distance Metric | Description |
| Euclidean |  |
| Manhattan |  |
| Minkowski |  |
| Chebyshev |  |

We now turn to an example that considers three pieces of information about a group of people - height in inches, weight in pounds, and age in years – so that each person is represented in 3-dimensional space with the vector *(height, weight, age)*. The Euclidean distance between individuals **A** = (74, 180, 17) and **B** = (66, 115, 33) is:

Because all the metrics treat the distance between each feature in an identical fashion, NN algorithms perform better when all data features have the same scale. Normalizing each of the feature variables in the pre-processing step is therefore important before training the classifier.

Another example is presented in Exhibit 11. The left-hand panel shows a scatter plot representing a set of training data in 2-dimensional space, with the color of each point representing that point’s class (red or blue). The purple *x* represents a new, unclassified point within the test data. The right-hand panel illustrates how the NN classifier finds the *k=5* closest points using the Euclidean distance. Of these five nearest neighbors, three are red and two are blue.

Exhibit 9: Alternative Distance Metrics for Nearest Neighbor Methods



NN classifiers are intuitive, easy to implement, and quick to train because they only need to store the training data. However, prediction can be computationally intensive because of the need to calculate the distance between the new data point and each point in the training data. If the training data is especially large, with many features, the NN algorithm might not be a viable machine learning option without first employing dimensionality reduction techniques such as PCA and feature selection.

Finally, the number of neighbors will affect the bias and variance of nearest neighbor classifiers. If *k* is too low, the classifier will have a high variance because a prediction will be decided by a small number of neighbors. This flaw might not show up during testing, but will be apparent once the classifier is used with new data. On the other hand, a high *k* value will increase bias because the neighbor groups will become too large, thereby incorporating points that should not be included. Choosing the right value for is important for improving a classifier’s performance.

## 5.2 Logistic Regression

**Logistic regression** models the log-odds of the occurrence of an event (in this case participation status), i.e. where is the probability of occurrence. The full model is

where represents the predictable part of the latent log-odds , and is a random error term with logistic distribution that represents the unpredictable part of . In the data, we observe the binary outcome , the actual participation status, instead of the latent log-odds which is a function of , the unobserved *ex ante* probability to participate. The log-odds converts the probability from its domain to the *logit* term with its domain over the entire real line. Upon finding the optimizing , the predicted *probability* of event occurrence is

for each observation in the sample. Since the domain of the exponential term in the denominator is , the predicted probability will always stay within the theoretical interval. The model then can freely optimize for parameter without the issue of making out-of-domain predictions, which would often occur with linear probability models that directly model and predict the probability term .

## 5.3 Naïve Bayes

Naïve Bayes methods are among the oldest, simplest and most popular machine learning classifications techniques. Their popularity stems from being extremely fast and surprisingly accurate given the strong assumptions. Because of these properties they are often employed as a baseline method for classification problems.

Naïve Bayes methods refer to a family of classifiers based on applying Bayes' theorem with strong independence assumptions between the features. In our classification problem, the probability of participating for a given set of physician-practice “features” is denoted and can be written using Bayes formula as follows

Deciding on the appropriate value of y is based on the quantity , which is defined as follows

If we guess that the the features X come from a physician that did submit MIPS data (. We can easily estimate the quantities and by taking the respective proportion of each outcome in the training data set, which are about 0.85 and 0.15, respectively.

The central challenge of Naïve Bayes algorithms is therefore to estimate the two functions . If X contains multiple variables, then represents a joint distribution which can require extremely large datasets and significant computational resources to estimate. Naïve Bayes algorithms drastically simplify this estimation process by assuming that each of the features are independent and therefore

so we can estimate the conditional probability distribution of each feature separately. In simple terms, a Naive Bayes classifier assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature.

The many variations of Naïve Bayes algorithms differ according to how they model the unknown functions. The simplest method, which is used when the feature variable is numerical, is to assume that the conditional probability follows a Gaussian distribution so the estimation procedure simply involves estimating the mean and variance of the data for the class and the class separately. In the quintessential Naïve Bayes application of text classification, it is assumed that the features represent counts, which are modelled as multinomial distributions.

Because our dataset contains both categorical and numerical data, as detailed in Exhibit 1, we trained both the Multinomial method on the categorical data and the Gaussian method on the continuous data. The Multinomial approach significantly outperformed the Gaussian approach (which is perhaps unsurprising given that most variables are categorical) so we report the results only for the Multinomial approach.

## 5.4 Random Forest

We first introduce the *decision tree classifier*, which is the basis of the random forest method.

A decision tree classifier is a graph of nodes where each node is associated with a data feature and a set of values related to that data feature. When an observation is passed through the tree, a “decision” is reached at each node based on its feature and associated values, which determines the child node the data point is passed onto. For example, in the case when there are only two child nodes, the algorithm compares the value of the specified feature of the sample data with the node’s value. If the sample’s value is less, the algorithm continues to the node’s left child. Otherwise, it goes to the right child. This process continues at each node until a leaf is reached, which contains a class label that is used to classify the data sample.

A simple decision tree classifier is trained by testing every possible feature and value combination on each node. A common approach to this is through the use of the *Gini Impurity* (GI)value, which measures the quality of a binary separation induced by a node. For data set X with samples and classes, the GI of the set is

where is the percentage of samples belonging to class in the dataset. A lower GI value means that the samples in the left and right set mostly belong to one class. A value of 0 means that both sets are pure, containing only one unique class. The algorithm finds the best possible feature and value combination to minimize the GI value of the current data set. The algorithm then recursively performs the same operation on each child node until a set consists of samples of only one class, which is then used to form a leaf node.

Exhibit 9 presents an example of a decision tree classifier in action on a two-feature dataset of height *h*, measured in inches, and weight *w*,measured in pounds. The algorithm first divides that data according to whether the value of *h* is greater than or less than 69. The GI of this separation can then be calculated as follows:

It turns out that 2/9 is the best attainable GI for this data set, so a node is therefore created with *feature* = and *value* = 69.

Exhibit 10: Decision Tree Classifier Example

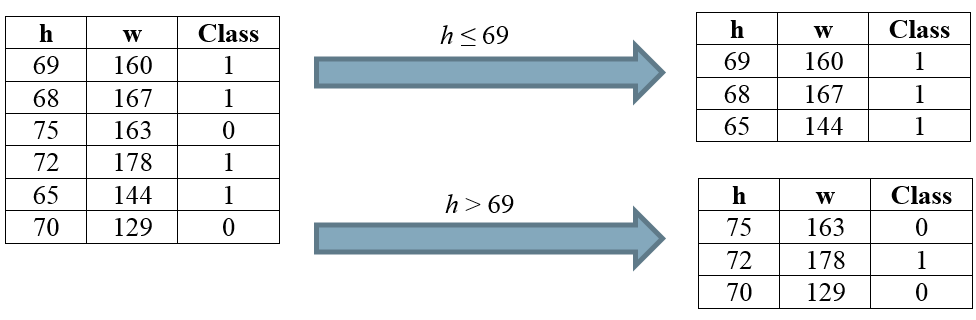
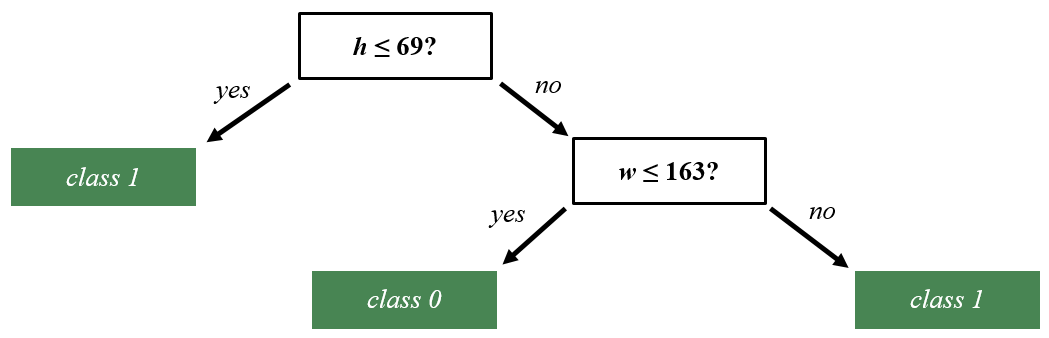


Exhibit 10 presents the optimal decision tree for the data presented in Exhibit 9. Because the data with all belong to class 1, the left-hand child of the parent node is a leaf. For the right-hand node, the classes can be divided using *feature* = and *value* = 163, which completes the decision tree. Consider an unclassified sample with features *h* = 71, *w* = 152. The decision tree predicts that this sample has *class\_value* = 0.

Exhibit 11: Decision Tree Classifier Example



A *Random Forest* classifier is an ensemble-learning model which aggregates multiple “weak learning” methods to create a strong classifier. The weak learning methods are usually simple decision trees. Random Forest methods create many individual decision trees with different subsets of the training data, and where each node of each decision tree is split using a randomly selected attribute from the data. Each tree in the forest predicts the class of a new data sample, and the class that is predicted by the most trees is ultimately predicted by the forest. Random forest methods have increased bias but also decreased variance. Overall, they tend to see an improvement in accuracy over simple decision trees.

## 5.5 Kernel Ridge Regression

Ridge regression methods are similar to traditional least-square methods but where the estimated coefficients are “shrunk” toward zero. More formally, the estimated coefficients are defined as follows

where the first term in the expression is the standard loss function in a least-squares problem and the second term is a penalty function with a tuning parameter that controls the strength of the penalty.

Ridge regression was initially introduced as a technique for analyzing multiple regression data that suffer from multicollinearity (as is common in many machine learning applications). When multicollinearity occurs, least squares estimates are unbiased, but their variances are large so they may be far from the true value. By adding a degree of bias to the regression estimates, ridge regression reduces the standard errors in the hope that the net effect will be to give estimates that are more reliable.

A Kernel Ridge Regression is simply a Ridge Regression where the feature variables are first converted into a higher-dimensional object so that the optimization problem becomes

where is a kernel that can be thought of as an inner product of combinations of features. Without going into the detail behind the mechanics, this formulation allows for a Richard feature space while also maintaining efficient computation.[[11]](#footnote-12)

# Chapter 6. Validation

There are many moving parts in this model, each of which has the potential to cause significant errors in the model output. This section details the various validation procedures we performed to ensure robustness of results.

## 6.1 FMLA Internal Validation

## 6.2 Robustness Checking

## 6.3 Sensitivity Checks

## 6.4 Standard Error Estimation

# Chapter 7. Conclusion

# Bibliography

XXX

# Appendix A. PArameter DictionarY – R Model

Needs to be updated

| **GUI Input Label** | **Location in GUI** | **Parameter** | **Description** |
| --- | --- | --- | --- |
| Imputation Method | General | impute\_method | method for imputation. |
|  |  |  |  |
| Leave Program | General | leaveprogram | Presence or absence of leave program |
|  |  |  |  |
| Wage Replacement Ratio | Program | base\_bene\_level | proportion of pay received as part of program participation |
| Benefit Effect | Behavior | bene\_effect | Whether to apply simulation of behavioral cost to applying to state program |
|  |  |  |  |
| Topoff Rate | Behavior | topoff\_rate | proportion of employers engaging in top-off substitution of paid leave with program benefits |
| Topoff Minimum Length | Behavior | topoff\_min\_length | Min length of leave required for top-off behavrior |
| Weekly Dependent Allowance | Program | dependent\_allow | weekly dependent allowance for those with children |
| Needers Fully Participate | Behavior | full\_particip\_needer | whether or not leave needers always take up benefits |
|  |  |  |  |
| Own Health | Behavior | own\_uptake | user-supplied benefit uptake rate for a given type of leave. |
| Ill Spouse | Behavior | illspouse\_uptake | user-supplied benefit uptake rate for a given type of leave. |
| Ill Child | Behavior | illchild\_uptake | user-supplied benefit uptake rate for a given type of leave. |
| Ill Parent | Behavior | illparent\_uptake | user-supplied benefit uptake rate for a given type of leave. |
| Maternity Disability | Behavior | matdis\_uptake | user-supplied benefit uptake rate for a given type of leave. |
| Bonding | Behavior | bond\_uptake | user-supplied benefit uptake rate for a given type of leave. |
| Waiting Perioid | Program | waiting\_period | how long in working days must leave takers wait to claim leave benefits |
| Clone Factor | Advanced | clone\_factor | Create clones of ACS records |
| Base Extension Effect | Behavior | ext\_base\_effect | Whether to apply base leave extension behavior in presence of program. standard leave extension effect from ACM model |
|  |  |  |  |
| Extend Probability | Advanced | extend\_prob | additional leave extension effect: probability of extension |
| Extend Days | Advanced | extend\_days | additional leave extension effect: fixed days of extension |
| Extend Proportion | Advanced | extend\_prop | additional leave extension effect: proportionate extension |
| Own Health | Program | maxlen\_own | max number of leave type days benefits can be claimed in a year |
| Ill Spouse | Program | maxlen\_illspouse | max number of leave type days benefits can be claimed in a year |
| Ill Child | Program | maxlen\_illchild | max number of leave type days benefits can be claimed in a year |
| Ill Parent | Program | maxlen\_illparent | max number of leave type days benefits can be claimed in a year |
| Maternity Disability | Program | maxlen\_matdis | max number of leave type days benefits can be claimed in a year |
| Bonding | Program | maxlen\_bond | max number of leave type days benefits can be claimed in a year |
| Bonding or Ill Relative | Advanced | maxlen\_DI | max number of bond, ill relative leave days benefits can be claimed in a year |
| Own Health or Maternity Disability | Advanced | maxlen\_PFL | max number of matdis, own leave days benefits can be claimed in a year |
| Total | Advanced | maxlen\_total | max number of total days benefits can be claimed in a year |
| Weekly Benefit Cap | Program | week\_bene\_cap | max weekly benefits that can be collected |
| Weekly Benefit Cap Proportion | Program | week\_bene\_cap\_prop | option to cap max weekly benefits that can be collected at a proportion of the mean weekly wage |
| Weekly Benefit Minimum |  | week\_bene\_min | min weekly benefits that can be collected |
| FMLA Protection | Behavior | fmla\_protect | Indicates whether or not leaves that are extended in the presence of a program that originally were less than 12 weeks in length are constrained to be no longer than 12 weeks in the presence of the program |
|  |  |  |  |
| Annual Earnings | Program | earnings | earnings in dollars in past 12 months |
| Usual Weeks Worked | Program | weeks | weeks worked in past 12 months |
| Usual Hours Worked | Program | ann\_hours | total number of hours worked in past 12 months |
| Minimum Firm Size | Program | minsize | Number of employees working at their employer |
| Weight Factor | Advanced | weightfactor | Multiply ACS weights by a certain number |
| Random Seed | Advanced | random\_seed | set random seed if user wishes analyses to be replicable |
|  |  | SELFEMP | Whether to include self employed workers in ACS data set |
|  |  |  |  |
|  |  | FEDGOV | Whether to include gov't workers in ACS data set |
|  |  |  |  |
|  |  | STATEGOV | Whether to include gov't workers in ACS data set |
|  |  |  |  |
|  |  | LOCALGOV | Whether to include gov't workers in ACS data set |
|  |  |  |  |
|  |  | formula\_prop\_cuts | Specification for formulaic benefits based on state mean wage |
|  |  | formula\_value\_cuts | Specification for formulaic benefits based on absolute wage values |
|  |  | formula\_bene\_levels | Proportion of pay those under each cut receive |
|  |  | elig\_rule\_logic | Description of the logic used when multiple eligibility criteria are specified. |
|  |  | own\_elig\_adj | user-supplied eligibility adjustment for a given type of leave. |
|  |  | illspouse\_elig\_adj | user-supplied eligibility adjustment for a given type of leave. |
|  |  | illchild\_elig\_adj | user-supplied eligibility adjustment for a given type of leave. |
|  |  | illparent\_elig\_adj | user-supplied eligibility adjustment for a given type of leave. |
|  |  | matdis\_elig\_adj | user-supplied eligibility adjustment for a given type of leave. |
|  |  | bond\_elig\_adj | user-supplied eligibility adjustment for a given type of leave. |

# Appendix B. Libraries Dependencies– R Model

| **Library** | **Version** | **Model’s Use of Library** |
| --- | --- | --- |
| stats |  | General tools for data analysis; used for logit regression |
| rlist |  | General tools for list manipulation |
| MASS |  | Ordinal regression function (polr) |
| plyr |  | General tools for data manipulation |
| dplyr |  | General tools for data manipulation |
| survey |  | Tools for using survey weights in logit regression |
| class |  | Alternate KNN function |
| dummies |  | Function for creating dummy variables |
| varhandle |  | Reformatting factor variables |
| oglmx |  | Alternate ordinal regression function |
| foreign |  | Writing data files in additional formats foreign to R |
| ggplot2 |  | Graphics generator |
| reshape2 |  | Flexibly restructure and aggregate data |
| e1071 |  | Naïve Bayes classifier |
| pander |  | Formatting plain text tables |
| ridge |  | Ridge regression for classifier |
| DMwR |  | SMOTE technique |

1. Leave benefits: Access, Civilian Workers, National Compensation Survey, March 2016. Retrieved from

   https://www.bls.gov/ncs/ebs/benefits/2016/ownership/civilian/table32a.htm [↑](#footnote-ref-2)
2. US Department of Labor, Women’s Bureau. N.d. Paid Leave Analysis Grant Program. Retrieved from

   https://www.dol.gov/wb/media/paidleavegrants.htm. From 2014 through 2016, over $3 million were awarded through this program to states and municipalities. See the above link for the grantees. [↑](#footnote-ref-3)
3. This model uses data from the latest wave from 2012. DOL conducted two previous waves in 2000 and 1995 as well. [↑](#footnote-ref-4)
4. Previous waves were conducted in 2000 and 1995. The full technical report from the 2012 wave is available here: <https://www.dol.gov/asp/evaluation/fmla/FMLA-2012-Technical-Report.pdf> [↑](#footnote-ref-5)
5. [ACM Documentation citation] [↑](#footnote-ref-6)
6. 1,322 respondents indicated taking at least one leave. Of these, 96% of responses indicated one of these six reasons. 413 respondents indicated needing but not taking at least one leave. Of these, 94% were for one of these six reasons. There were three other categories of responses: “Address issues arising from military deployment”, “Other relative”, and “Other”. [↑](#footnote-ref-7)
7. This is not a trivial proportion of the respondents. 76 of 986 leave takers indicated they had 3 or more reasons for taking leave in the past 12 months in A4a. [↑](#footnote-ref-8)
8. Only 4 of 251 leave needer respondents indicated needing leave for four or more reasons in Q5b. [↑](#footnote-ref-9)
9. 672 of the 1551 FMLA leave takers/needers are considered to be financially sensitive by our composite variable. [↑](#footnote-ref-10)
10. [https://factfinder.census.gov/faces/nav/jsf/pages/searchresults.xhtml?refresh=t#](https://factfinder.census.gov/faces/nav/jsf/pages/searchresults.xhtml?refresh=t) . Retrieved 1/22/2019. [↑](#footnote-ref-11)
11. See Saunders, C., A. Gammerman and V. Vovk (1998). Ridge regression learning algorithm in dual variables. [↑](#footnote-ref-12)