**Overview**

These specifications describe programs and processes used to calculate provision of low value services (LVS). The LVS measures were developed by researchers at Harvard. We used code made publicly available at [this website](https://dataverse.harvard.edu/dataset.xhtml?persistentId=doi%3A10.7910%2FDVN%2FDEW0UO).

The programs and processes described here were originally used at Mathematica to calculate LVS provision among Medicare beneficiaries (using Medicare claims and administrative data) who were attributed to Comprehensive Primary Care (CPC) Classic and Comprehensive Primary Care Plus (CPC+) treatment and comparison practices. The purpose of these specifications is to provide guidance for development of a claims-based, beneficiary-level analytic file(s) to anyone looking to track provision of LVS among groups of providers and beneficiaries. For example, the measures team working on Mathematica’s evaluation of CMS’s Primary Care First (PCF) model are planning to track LVS among beneficiaries attributed to PCF intervention and comparison practices.

Many of the programs referenced in these specifications reflect the Harvard code. Programmers at Mathematica created several programs to convert ICD-9 to ICD-10 codes (025\_icd9\_icd10\_xwalk.sas), to create three new LVS (030\_flags\_newmeasures.sas), as well as to output unadjusted counts of LVS (060\_unadjust\_lvs\_counts\_sas).

The LVS measure set consists of 31 measures tracking various services. We have tried to replicate how Harvard specifies each of the 31 measures based on the code Harvard made available. For plain text references specifying each of the 31 measures, please see the 2015 paper by Schwartz et al ([link](https://jamanetwork.com/journals/jamainternalmedicine/fullarticle/2442504#:~:text=In%20year%201%20of%20Pioneer,41.0%20services%20per%20100%20beneficiaries.)) or visit the Harvard website linked above.

Note: Harvard specified these measures when ICD-9 was the prevailing coding regime. We have since updated the code to reflect the switch to ICD-10 coding. This was done using General Equivalence Mappings. Information on this is available [here](https://www.cms.gov/Medicare/Coding/ICD10/2018-ICD-10-CM-and-GEMs).

**Specifications**

These specifications outline the process of creating beneficiary-level output files. The beneficiary-level files are created using claims-level files. The program names used below are those used for the CPC+/PCF LVS team’s work from 2018-2021.

1. **Create Finder file**. This step depends on requirements for the given program and set of beneficiaries and providers being evaluated or analyzed. For the CPC work, the LVS team used a dataset of CPC practices and a crosswalk of beneficiaries attributed to CPC treatment and comparison practices. For example, for PCF analyses, one would include relevant PCF practice IDs and beneficiary IDs for the finder file.
2. Include the measurement period of interest (e.g., CY2020) and a lookback period (e.g., the year prior to the measurement period) for purposes of identifying beneficiaries eligible for inclusion in the dataset.
3. **To be included in the analysis file(s) Mathematica utilized, Medicare beneficiaries must have:**
   * Been continuously enrolled in Medicare fee-for-service (FFS) Parts A and B for the measurement year and preceding year,
   * Been Alive at the end of the analysis period,
   * Not been enrolled in an HMO, and
   * Had Medicare as the primary payer across the measurement and preceding periods.[[1]](#footnote-1)

**The output dataset following completion of steps 1-3 is named finder\_benes.**

1. **Merge Master Beneficiary Summary File (MBSF) data for beneficiaries included in the Finder file to create the variables needed for the Harvard program**. **The program used on CPC Classic data is named 000\_harvard\_vars**. Variables pulled from the MBSF:

|  |
| --- |
| bene\_id, bene\_birth\_dt, age\_at\_end\_ref\_yr, sex\_ident\_cd, bene\_race\_cd, zip\_cd, bene\_hi\_cvrage\_tot\_mons, bene\_smi\_cvrage\_tot\_mons, bene\_hmo\_cvrage\_tot\_mons, mdcr\_entlmt\_buyin\_ind:, hmo\_ind:, bene\_death\_dt, county\_cd, state\_code, dual\_stus\_cd:, entlmt\_rsn\_orig, esrd\_ind |

1. **Use the Finder file to pull all beneficiary claims from the Medicare Outpatient, Carrier/Part B, and MedPAR files for the years of interest, and create relevant analysis file(s).[[2]](#footnote-2) The associated program names are 001\_opd\_claim\_pull, 002\_ptb\_claim\_pull, and 003\_medpar\_claim\_pull.**

Below are the variables pulled from each file (based on documentation from Dean Miller):

* + **Outpatient**

|  |
| --- |
| Pulls diagnosis codes, revenue center codes, provider number, and claim from and thru dates from outpatient claims for the individuals in Finder file for use in Harvard’s LVS programs. |

|  |
| --- |
| Pulls claims header file, Rif&year..outpatient\_claims\_&month.  Pulls variables clm\_id, clm\_from\_dt, clm\_thru\_dt, prvdr\_num, prncpal\_dgns\_cd, icd\_dgns\_cd1 through icd\_dgns\_cd25 |

|  |
| --- |
| Pulls claim-line file for revenue. Rif&cyear..outpatient\_revenue\_&month.  Pulls variables bene\_id, clm\_id, hcpcs\_cd, rev\_cntr\_dt, rev\_cntr, and various rev\_cntr amount variables. |

|  |
| --- |
| Output files are opc&yr. and opr&yr. |

* + **Carrier/Part B**

|  |
| --- |
| Pulls diagnosis codes, betos codes, hcpcs, and other relevant information from Part B claims for the individuals in finder\_benes for use in Harvard’s LVS programs. |

|  |
| --- |
| Pulls from rif&year..bcarrier\_claims\_&month.  Variables are bene\_id, clm\_id, clm\_from\_dt, clm\_thru\_dt, nch\_near\_line\_rec\_ident\_cd, carr\_cm\_pmt\_dnl\_cd, nch\_clm\_type\_cd, prncpal\_dgns\_cd, icd\_dgns\_cd1 to icd\_dgns\_cd12 |

|  |
| --- |
| Pulls from rif&year..bcarrier\_line\_&month.  Variables are bene\_id, clm\_id, clm\_line\_num, betos\_cd, line\_1st\_expns\_dt, hcpcs\_cd, line\_place\_of\_srvc\_cd, line\_alowd\_chrg\_amt, prf\_physn\_npi, org\_npi\_num, tax\_num, prvdr\_spclty, line\_icd\_dgns\_cd |

|  |
| --- |
| Output datasets are carl&yr. and carc&yr. |

* + **MedPAR**

|  |
| --- |
| Pulls data for MedPar files.  Variables are: bene\_id, admsn\_dt, dschrg\_dt, drg\_cd, srgcl\_prcdr\_1\_cd - srgcl\_prcdr\_6\_cd, ip\_admsn\_type\_cd, src\_ip\_admsn\_cd, er\_chrg\_amt, icu\_ind\_cd, mdcr\_pmt\_amt, bene\_pta\_coinasrnc\_amt, bene\_ip\_ddctbl\_amt, bene\_blood\_ddctbl\_amt, bene\_prmry\_pyr\_amt, medpar\_yr\_num, prvdr\_num, admtg\_dgns\_cd, admtg\_dgns\_vrsn\_cd, dgns\_vrsn\_cd\_1-dgns\_vrsn\_cd\_25, dgns\_1\_cd - dgns\_25\_cd, ss\_ls\_snf\_ind\_cd |

|  |
| --- |
| Output dataset is medpar\_&yr. |

1. **Run program named 020\_mcareextracts.sas on analysis file(s) created above.** As part of the claims pull, the program pulls a claim header file and a claim line file. With this program, we merge the claim header and line files with the MBSF file as well as our finder file to the final claims dataset searched in the 030\_flags.sas program in step 8.
2. At this point, the LVS team updated the Harvard code to account for ICD-10 codes. We ran a program named “readin\_lvs\_eligwts” to create a SAS dataset out of a spreadsheet which crosswalks the LVS ICD-9 codes to ICD-10 codes.[[3]](#footnote-3)
   * Next, the LVS team ran a program named “025\_icd9\_icd10\_xwalk” on the VRDC. This program adds flags to the SAS dataset for each LVS inclusion and exclusion list to indicate whether a specific ICD-9 code was included in a list for a given LVS measure.[[4]](#footnote-4)
   * Following this step, the LVS team added code to the beginning of relevant Harvard programs (currently programs 030 and 035) that uses the crosswalk data to create macro lists of ICD-10 codes that are then inserted into the original Harvard code.[[5]](#footnote-5)
3. **Run program named 030\_flags.sas on analysis file(s).** This program searches for LVS from claims. It creates two datasets: one has LVS identified using their “specific” detection criteria (denoted by the word “small” or abbreviation “sm”) and one has LVS identified using their “sensitive” detection criteria (denoted by the word “large” or abbreviation “lg”). **We have used the “small” dataset.**
   * The Harvard code assigns names or “flags” for each of the 31 measures, and it also assigns procedure and condition codes for each flag.
   * The program also creates exclusionary datasets. Using the PTH measurement for patients with stage 1-3 chronic kidney disease (CKD) on dialysis (PTH\_6) as an example, the program first pulls all claims with procedure code 83970 with an expense date after the CKD diagnosis. Next, the program identifies all claims with the exclusionary diagnosis codes. The third step uses the exclusionary claim set to find the first date of dialysis in the current or lookback period. Finally, the program merges the date of first dialysis into the inclusionary set and keeps only those claims with an expense date after the first date of dialysis.
4. **Run program named 030\_flags.newmeasures.sas on analysis file(s).** As part of our update of the Harvard measure set, we removed 3 of the original measures (preoperative chest radiography, inferior vena cava filters for prevention of pulmonary embolism, and arthroscopic surgery for knee osteoarthritis) and added 3 new measures (Laminectomy or spinal fusion, electromyography (EMG) for low back pain, and magnetic resonance imaging of the peripheral joints to monitor rheumatoid arthritis. The variable prefixes for the 3 new measures are ‘lam’, ‘mri\_ra’, and ‘emg’, respectively.

**Note:** The specifications as reflected in the programming code posted to GitHub for the three new measures Mathematica created include a “look forward” period for associated claims data. The other 28 measures in the updated measure set do not include claims from the “look forward” period. The “look forward” period would be the year following the measurement or observation year of interest.

1. **Run program named 035\_covars.sas on analysis file(s).** This file constructs beneficiary-level covariates from the beneficiary summary file and claims for each year of data. It merges covariates to beneficiary sample files (**out.ourbenes ‘year’**) and can combine years to create a single beneficiary-year-level file (**out.ourbenescovars**).
   * This program also creates the “denominator flag” variables, which have the prefix “dfl\_”. These indicate whether a patient qualified for potential receipt of a low-value service (e.g., patients who had a diagnosis of plantar fasciitis qualify them for potentially receiving low-value imaging for plantar fasciitis).
   * DFL flags are based on having one or more claims with a specific combination of betos, hcpcs\_cd, and/or ICD diagnosis codes.  The specific combination differs for the different dfl\_ flags.
   * The LVS team found that the Harvard denominator (“dfl\_”) flags do not always represent the groups of beneficiaries who qualify for a given measure.[[6]](#footnote-6) As such, the LVS team created eligibility variables and descriptions, populated in an Excel workbook with a tab named ‘Dfl\_Eli\_Variables’. These variables are supposed to indicate whether a beneficiary qualifies for a given measure’s denominator and can be found in the program named 060\_unadjust\_lvs\_counts.sas.
2. **Run program named 040\_flags2.sas.** Per the Harvard documentation, this program:
   * Combines and de-duplicates all low-value services from flags.sas, by year.
   * Merges combined/de-duplicated low-value services with standardized prices to create spending variables for each low-value service.[[7]](#footnote-7)
   * Calculates each beneficiary’s number of low-value services received, by year. Appends all years of beneficiary’s low-value use together.
   * Merges beneficiary counts with beneficiary sample dataset (out.ourbenescovars) and with additional covariates, to create a final analytical dataset (out.analysis)
3. **Run program 045\_covars2.** 
   * Adds practice, treatment flag, and number of beneficiaries per practice to the data file produced by 040\_flags2 (yranalysis).[[8]](#footnote-8) The added variables are pulled from the initial finder file created by program 000 (finder\_benes).
   * Output dataset = yranalysis2
4. The LVS team also has a program which produces unadjusted practice-level rates for the each of the 31 LVS measures and a total unadjusted rate. That program is named **060\_unadjust\_LVS\_counts** and it produces **measure counts per 100 beneficiaries per practice** for a given measurement period.
5. **To calculate rates by measure and overall** (across the 31 LVS measures) for each practice, we followed the Harvard method and present service rates as the number of services per 100 beneficiaries who qualified for inclusion in the dataset (i.e., **beneficiaries who met the criteria in Step 3 of the Specifications section**).

1. These requirements were specified in the Harvard code and what the LVS team used for its analyses. [↑](#footnote-ref-1)
2. Note: Harvard’s code in some cases looks through as many as 2 years of claims prior to an LVS on a given claim to verify whether a service should be included in numerator counts. [↑](#footnote-ref-2)
3. Each row in the dataset contains an ICD-9 code used in the Harvard code and the corresponding ICD-10 flag. [↑](#footnote-ref-3)
4. Taking LVS measure with flag #11 (vertebroplasty or kyphoplasty for osteoporotic vertebral fractures) as an example, any claims with an ICD-9 code that started with ‘73313,’ ‘8052,’ or ‘8054’ were flagged as claims that count towards this measure count. This code scans the crosswalk for any ICD-9 codes that meet this requirement and assigns these rows a value of ‘1’ on flag cond11. The final crosswalk dataset is then output as “lvs\_icd\_xwalk\_fin.” [↑](#footnote-ref-4)
5. For example, the original Harvard code for the LVS measure #11 flags a claim for inclusion if any of the diagnosis codes on the claim meet the following requirement: index(dgnsall, ' 73313 ') + index(dgnsall, ' 8052 ') + index(dgnsall, ' 8054 ') > 0. For the update, a macro list of all ICD10 codes that match the original ICD9 code is created (for LVS 11 it is named &cond11). We then flag claims for inclusion if any of the diagnosis codes on the claim meet the following requirement: in(&cond11.). Thus, it is not the exact same code as the Harvard code, but the steps are the same in terms of how claims are flagged. [↑](#footnote-ref-5)
6. For example, the denominator flag (‘dfl\_sync’) for LVS measure #22 (screening for carotid artery disease for syncope) identifies individuals with a diagnosis of syncope; however, the measure excludes beneficiaries with a history of stroke (CCW indicator of stroke, i.e., ‘ccw\_stroke’ = 1). As such the LVS programming team included that requirement to define the associated eligibility variable (‘eli\_ctdsync’) for this LVS measure. [↑](#footnote-ref-6)
7. We do not use Harvard’s costs.sas program which constructs standardized prices for each LVS. This part of the code is not used. In the flags2.sas program, this part of the code begins with **<merge in pricing info from costs, collapse by bene>.** [↑](#footnote-ref-7)
8. This is the point at which the LVS programs incorporated CPC-specific variables. This program would need to be edited to accommodate other datasets, variable names, etc. [↑](#footnote-ref-8)