**Chronic Conditions Package User Guide**  
By Patricia Ferido

The purpose of the Chronic Conditions Package is to analyze claims data and flag the presence of chronic conditions based on user-defined criteria. There are three main criteria for identifying chronic conditions in the claims data: qualifying diagnosis codes, location of diagnosis codes (e.g. inpatient, outpatient, skilled nursing facility, etc.), and the reference period in which to find the diagnoses e.g. 2 years, 3 years). The final outputs will be yearly beneficiary level files with monthly condition and enrollment flags.

Condition flags will have one of the four following values:

|  |  |
| --- | --- |
| .I | Insufficient data for analyzing criteria |
| 0 | No insurance coverage or qualifying claims |
| 1 | Qualifying claims, but no coverage |
| 2 | Qualifying coverage, but no claims |
| 3 | Qualifying coverage and claims |

The current set up will identify the 27 conditions currently part of the Center for Medicare/Medicaid Service’s Chronic Condition Warehouse as of the August 2017 revision. Researchers can customize the input files to look for any conditions of interest. Below are detailed instructions for how to use the package correctly. All instructions are crucial for the macro to work. For an abridged version, see the Checklist.

1. Create a copy of the “Chronic\_Conditions\_Package” Folder (and all its contents) into the location of the project where you intend to work. DO NOT edit or run the “Chronic\_Conditions\_Package” in its source location, as this is a resource shared by many people. If you wish to use the default CMS algorithms, skip to step 3, otherwise if you wish to customize the condition algorithms, read ahead.

**Customizing Condition Algorithms**

1. Navigate to the “csv\_input” folder, where you will find 3 CSV files that you can customize to determine your conditions of interest and their corresponding algorithms.
   1. **CC\_codes** – This file lists all of the diagnosis codes that will identify your conditions. By default, it is set up to identify the CCW conditions using the ICD9 and ICD10 codes.

Variables

**Condition** - Shorthand condition name, i.e. ALZH = Alzheimer’s Disease.

* Please create a shorthand name that’s less than 10 characters

**CodeType** – This column specifies the type of code used to identify the condition and can only take the following values:

* ICD9DX – ICD-9 Diagnosis Codes, used before September 30, 2015
* ICD10DX – ICD-10 Diagnosis codes, used from October 1, 2015 on
* ICD9PRCDR – ICD-9 Procedure Codes, used before September 30, 2015
* ICD10PRCDR – ICD-10 Procedure Codes, used from October 1, 2015 on
* HCPCS – HCPCS/CPT Procedure Codes

**DxCodeLocation -**  This column specifies which diagnosis/procedure code variables will be considered and can only take the following values:

* ANY – any of the codes
* DX1 DX2 – only the primary or secondary codes
* DX1 – only the primary code will be considered

As a note, the program will assume that there will be a one-to-one relationship between ‘dxcodelocation’ and ‘condition’. For example, all atrial fibrillation codes, condition=‘ATF’, have to be found in ‘DX1 DX2.’ If you want to be more specific and have a separate location for different codes within a condition, you will have to create different condition names. For example, instead of just having ‘ATF’, you may want to create ‘ATF1’ where all codes associated with that condition must be in ‘DX1 DX2’ and also ‘ATF2’ where all codes associated with that condition can be in ‘ANY’.  
**DxCode** – This column is the actual diagnosis code.

* + - This variable must be formatted as a character variable and should be written exactly as shown in the CCW documentation
    - If there is a leading zero in the documentation, then include any leading zeroes and any periods.
  1. **CC\_Desc –** This file is crucial for setting up your algorithm and conditions of interest. You will need to have in mind the type of claims (ex: IP, SNF, HHA), the number of claims, the reference period, and the days between claims that will identify your condition in the data. For example, the CCW requires that 1 Carrier claim is necessary to be diagnosed with Glaucoma. The default set up uses the CCW Condition Algorithms from the August 2017 revision.

Variables

**Condition –** This is the same as the condition variable in *CC\_codes*. It is the shorthand version of the condition name.

**Condition\_long** – This is the full name of the condition referenced by the *Condition* variable

**Claim\_type1 and Num\_DX1 –** The *claim\_type/num\_dx* variables go hand in hand to specify the type of claims and the number of this type of claim that need to be seen in the data to qualify for the algorithm. If you want to look within multiple claim types without discriminating between type, then list them all in the same variable separated by a comma. For example, if you want to find 2 outpatient and carrier claims and don’t care about seeing a specific combination, then put both separated by a comma. See the following:

|  |  |
| --- | --- |
| Claim\_type1 | Num\_dx1 |
| OP,CAR | 2 |

Important Notes:

* You can list as many *claim\_type* and *num\_dx* variables as you’d like. Make sure there are an equal number of *claim\_type* variables as there are *num\_dx* variables or the program will not run.
* You can list any type of claim you want, keeping in mind that you must create a variable called *claim\_type* in your claims data that will categorize the kind of claim listed. For example, if you want to look at ER claims and want to put *Claim\_type1* as “ER”, that is completely acceptable as long as you also have a way of coding for ER visits in your data.
* You can include any positive number of claims in the *num\_dx­* variables. Do not put any character values in these variables or the program will not run.

**Min\_days\_apart  *-***Minimum number of days between claims before they can be counted as a new claim. By default, the minimum days apart is set to 1 which would require the next claim to be at least 1 day from the previous qualifying one. The CCW uses the claim thru date to identify claims and this program replicates that method. If you would like to place a restriction on claims so that they are at least a week from each other, then *min\_days\_apart* would be set to 7. The date of the first claim is used as an anchor.

**Max\_days\_apart** – Maximum number of days between claims will be most useful for when you would like to exclude claims with long periods of time between each other. By default, the maximum days apart will be blank, but if you would like to require that claims be within 6 months of each other then *max\_days\_apart* would be set to 180.

**Ref\_months** – This column refers to the reference period in which claims will need to be found. The value must be in month units, which allows reference periods to be fractions of years if necessary.

**Source** – The documentation source for the algorithm. If the algorithm was customized, I would highly recommend keeping documentation elsewhere of the rules set up for each condition.

* 1. **CC\_Exclude** – This file lists all of the diagnosis codes that you might want to use to identify claims to exclude. For example, in the CCW there are a few diagnosis codes that signify that a claim should be disregarded even if there is a diagnosis for stroke in other diagnosis variables.

Variables

**Condition –** same shorthand condition as the previous two files

**CodeType** – This column specifies the type of code used to identify the condition and can only take the following values:

* ICD9DX – ICD-9 Diagnosis Codes, used before September 30, 2015
* ICD10DX – ICD-10 Diagnosis codes, used from October 1, 2015 on
* ICD9PRCDR – ICD-9 Procedure Codes, used before September 30, 2015
* ICD10PRCDR – ICD-10 Procedure Codes, used from October 1, 2015 on
* HCPCS – HCPCS/CPT Procedure Codes

**DxCodeLocation -**  This column specifies which diagnosis/procedure code variables will be considered and can only take the following values:

* ANY – any of the codes
* DX1 DX2 – only the primary or secondary codes
* DX1 – only the primary code will be considered

As a note, the program will assume that there will be a one-to-one relationship between ‘dxcodelocation’ and ‘condition’. For example, all ‘ATF’ codes have to be found in ‘DX1 DX2.’ If you want to be more specific and have a separate location for different codes within a condition, you will have to create different condition names. For example, instead of having just having ‘ATF’, you may want to create ‘ATF1’ where all codes associated with that condition must be in ‘DX1 DX2’ and also ‘ATF2’ where all codes associated with that condition can be in ‘ANY’.

**DxCode** – This column is the actual code.

* + - This variable must be formatted as a character variable and should be written exactly as shown in the documentation
    - If there is a leading zero in the documentation, then include any leading zeroes and any periods.

**Preparing Input Claims Data**

1. Prepare your input claims data sets
   1. You can input as many claims-level data sets as you wish to be included in the analysis. If your data comes in yearly or monthly files, they can be left as is as long as they follow the format and variable requirements described below. Keep in mind that the package will not do any checks on whether or not you’re including all the data necessary, so be careful to include all years and claims of interest. Keep in mind that if you want to measure a condition with a 3-year reference period, you will need to include at least 3 years of data including the year of interest and the 2 years prior. If you have both header and line claims data, only include the header files.
   2. Your claim data sets need to have the variables listed in the table below. Unless you are customizing the package to include procedure codes and only diagnosis codes are needed. The column on the left describes the variable type, the middle column lists what the variable names should be standardized to, and the rightmost column lists standardized format. In a few cases, no standard variable name is required and the variable name can be left as it is in the original source. Asterisked variables are required and will produce errors if left out of the data set or incorrectly formatted.

|  |  |  |
| --- | --- | --- |
| **Variable Type** | **Standardized Variable Names** | **Format** |
| Unique patient identifier\* | No standard name required | No standard format |
| ICD-9 Diagnosis Codes | Icd9dx1-icd9dx[max]\* | Character\* |
| ICD-10 Diagnosis Codes | Icd10dx1-icd10dx[max]\* | Character\* |
| ICD-9 Procedure Code | Icd9prcdr1-icd9prcdr[max]\* | Character\* |
| ICD-10 Procedure Code | Icd10prcdr1-icd10prcdr[max]\* | Character\* |
| HCPCS Procedure Codes | Hcpcs1-hcpcs[max]\* | Character\* |
| Claim date\* | Claim\_dt\* | Date\* |
| Claim type\* | Claim\_type\* | Character\* |

* 1. Diagnosis/procedure code variables need to be numbered according to priority. For example, if there is a primary ICD-9 diagnosis variable, then *icd9dx1* should be the primary diagnosis. If you have 10 other ICD-9 diagnoses then they will be named *icd9dx2-icd9dx11.*
     + Some data sets might not have icd10 codes or certain kinds of procedure codes, which is to be expected especially in earlier years. For now, it is important to make sure that variable names match the type and that priorities are correct.
     + You might have to do some processing to separate out ICD9 from ICD10 codes. Many claims data sets will have an “ICD Version” variable that will tell you whether or not the variable is ICD-9 or ICD-10.
  2. The claim date should be the date in your data that is the most reliable date for the claim and should be formatted as a date variable. If your data has both a claim start date and a claim end date, then select the more reliable one. For example, in the Medicare data, the “thru\_dt” is the most reliable date and the CCW algorithm doesn’t use the claim start date and so only the claim end date is necessary.
  3. If you want to replicate the CCW algorithm as much as possible, you will need to drop any diagnoses associated with ambulance services or durable medical equipment claims. The CCW algorithm excludes claims where the services do not require a licensed health care professional. This means diagnoses associated with BETOS codes D1A, D1B, D1C, D1D, D1E, D1F, D1G (which is DME), or O1A (which is ambulance services) should be dropped. In the Medicare data, only carrier claim types 71 and 72 are carrier claims of interest. Please note that if you do want to include Durable Medical Equipment Claims and/or Ambulance Services then do not do this restriction, but know that you will get different results from the CCW.
  4. The *claim\_type* variable will need to match the claim types that were defined in *CC\_Desc*. They are meant to be shorthand references to the different kind of claims of interest. You may want to use the same claim types from the CCW, or create your own set of claim types. Either is fine as long as the claim types match between *CC\_Desc* and your input claims data.
  5. Drop all other variables. This will speed up the process and simplify the output. If there is more information that you want to keep, you can merge it back later.
  6. Sort the claims data set by unique beneficiary id and claim\_dt.
  7. Save all cleaned claims data sets into one folder.

**Preparing Enrollment Data**

Enrollment data must be yearly, beneficiary level files.

1. Prepare your enrollment/insurance coverage data sets
   1. Along with identifying claims requirements, the CCW algorithm also flags insurance enrollment. Depending on your data source, you might already have the necessary enrollment files on hand. They are already available on Optum, NBER and on the VRDC. The files will be yearly beneficiary-level files with monthly variables indicating the number of months enrolled before and including the current month and the number of months enrolled after and including the current month.
   2. If already available, you may want to limit the enrollment data to a subset of beneficiaries, as the package will run the analysis for all beneficiaries in both the enrollment and claims data sets.
   3. If they are not already available, then pick between either creating either the annual or period shape below depending on the structure of your enrollment data. Macro variables defined in step 5 will be determined by which shape is chosen.

|  |  |  |
| --- | --- | --- |
| **Annual Shape:** Enrollment input files for each year with one record per beneficiary-month. Named as such - libref.[prefix][yyyy] | | |
| **Required Variables** | **Standardized Variable Names** | **Format** |
| Unique patient identifier\* | No standard, but should be same as input claims data sets | No standard format, but should be same as input claims data sets |
| First day of month\* (one for each month) | Date\* | Date format |
| Enrollment variable\* | No standard | Binary, 1=enrolled, 0=not enrolled |

|  |  |  |
| --- | --- | --- |
| **Period Shape:** Enrollment input files with one record per beneficiary and period of enrollment. It can have multiple records per beneficiary (i.e. if there is a gap in enrollment, the file can have a record for the first period and a record for the second period after the gap) | | |
| **Required Variables** | **Standardized Variable Names** | **Format** |
| Unique patient identifier\* | No standard, but should be same as input claims data sets | No standard format, but should be same as input claims data sets |
| Start of enrollment period\* | Begdt\* | Date format |
| End of enrollment period\* | Enddt\* | Date format |

* 1. The output continuous enrollment files will have the following variables:

|  |  |  |
| --- | --- | --- |
| **Variable Type** | **Standardized Variable Names** | **Format** |
| Unique patient identifier\* | No standard, but should be same as input claims data sets | No standard format, but should be same as input claims data sets |
| Months enrolled pre-month including reference month | [prefix]\_pre\_01-[prefix]\_pre\_12\* | Numeric\* |
| Months enrolled post-month including reference month | [prefix]\_post\_01-[prefix]\_post\_12\* | Numeric\* |

**Running the Package**

1. Navigate to the “programs” folder. Prepare the SAS file titled “input\_program” in the programs folder.
   1. The success of the package is contingent on the following macro variables being manually and accurately set. The program will issue error messages for any that are incorrectly set.
   2. The first thing will be to create your libname’s for any of the locations where your input or output data will be found
   3. The following macro variables will have to be input to the function manually:
      * *Projhome* – Copy and paste the file path of the project where you are running the macro, ending with the “Chronic\_Conditions\_Package” folder
      * *ID* – name of unique patient identifier variable.
      * *Minyear –* First year of data that you want to process
      * *Maxyear* – Last year of data that you want to process
      * *Claims\_data* – this should be the names of the cleaned claims data sets you are processing. If there are more than one data sets, list them separated by 1 space.

e.g. clmin.clean\_data1 clmin.clean\_data2 clmin.clean\_data3;

* + - * Names of input claims data sets cannot be 30 characters long or longer
      * Make sure to include fileref if inputting permanent files
    - *Create\_enr* ***–*** Y/N
      * Y – enrollment files need to be created by the package
        1. *create\_enr\_shape* - two possible inputs: A – for annual, P – for period (See Step 4 for more info on shape)
        2. *create\_enr\_filein -* provide input files of beneficiary enrollment information with libref
        3. *enr\_prefix* - name of the enrollment files you want to create with libref, yearly suffix will be added automatically
        4. *enr\_*var - name of enrollment variable you want to use (e.g. MA, FFS, enr), monthly suffix will be added automatically. If using A shape, this is the same var as the month enrollment flag (see Step 4 for info on shape)
      * N – enrollment files already exist and are not to be created by the package
        1. *Create\_enr\_shape* – leave blank
        2. *Create\_enr\_filein***–** leave blank
        3. *enr\_prefix* - name of the enrollment files you want to create with libref, yearly suffix will be added automatically
        4. *enr\_var* - name of enrollment variable you want to use (e.g. MA, FFS, enr), monthly suffix will be added automatically. If using A shape, this is the same var as the month enrollment flag (see Step 4 for info on shape)
    - *Claims\_out\_prefix* - prefix of yearly output data sets without the year suffix. Make sure to include a fileref if creating permanent files
    - *Custom\_algorithm –* Y/N, default is N
      * Y - if input excel sheets have been modified and custom algorithms should be used
        1. *Custom\_suffix* – If Y is specified above then this suffix will be added to created algorithm files and permanent files will be added to the ‘data’ folder for future reference. Default is ‘\_custom’.
        2. *Custom\_cond* – Default is to be left blank, which runs the package for all 27 conditions in the CCW list. Can be customized to pull only certain conditions by inputting shorthand names for conditions of interest separated by a space. Shorthand names to use can be easily found in the ‘CC\_Desc’ excel sheet.

e.g. DIABETES ALZH RAOA

* + - * N – if default CCW condition algorithms should be used
        1. *Custom\_suffix*  - leave blank
        2. *Custom\_cond* – leave blank

1. Run the “input\_program” program
   1. The success of this package is contingent on the above CSV files, your clean input claims, and enrollment data sets being formatted correctly. If there are any errors, the program will issue error messages in the log and terminate prematurely.
   2. Some checks to do before running:
      * Make sure you put the correct project file paths above
      * Make sure that the *condition* variable on any of the files is less than 10 characters long
      * Make sure that all conditions that are present on *CC\_Desc* are also on *CC\_Codes* and vice versa. Make sure that conditions on *CC\_Exclude* are also on the other files.
      * Make sure that all variables that require certain numeric values have no invalid values
      * Make sure that there are no alphabetic values in *Num\_dx* and *min\_days\_apart* and ­*Ref\_months* numeric variables
      * Make sure all the claims data sets have the correct variables and claim types
      * Make sure you are including all years of claims data and all types of claim data necessary for analysis
   3. Check log for further warnings and errors.
   4. If this program runs successfully, you will see the following data sets in your output folder and the variable values for the flags in the second table below. Please note that the output data will show every employee who is in the claims or the enrollment files. If an employee is in one source and not the other, they will still be in the output.

|  |  |
| --- | --- |
| **Name of Output Data Set** | **Description of Data Set** |
| [custom\_output\_name] [yyyy] | Yearly beneficiary-level files with monthly condition flags that flag whether or not the beneficiary had the condition under the definitions of the algorithm |
| Cc\_codes[custom\_suffix], cc\_exclude[custom\_suffix], cc\_desc[custom\_suffix] | If you customized an algorithm, these are permanent SAS data sets of your custom algorithm files with the suffix specified by the function |

**Values for Condition Flags**

|  |  |
| --- | --- |
| .I | Insufficient data for analyzing criteria |
| 0 | No insurance coverage or qualifying claims |
| 1 | Qualifying claims, but no coverage |
| 2 | Qualifying coverage, but no claims |
| 3 | Qualifying coverage and claims |

**Caveats**

There are a few reasons why the flags here won’t match exactly with the CCW.

* The CCW had access to data since 1999 so if the claims data that you feed into the program start later, be aware that the first few years may have a censoring issue due to the reference period. For example, if the Medicare data starts in 2002 that means conditions with a reference period of 3 years won’t be able to look back until 1999 and will only have data from 2002. The package will return a special missing value noting that the data is insufficient for analysis.
* In order to get the first ever condition date, the CCW uses the first time the condition qualifies for a mid or end year claim flag. This algorithm identifies the first ever condition date even if it is met outside of the mid or end of the month.