**MDS Algorithm Version 1:** In version 1, I append all years of the raw MDS data from 2006 to 2013 together. Then, from there I standardize the names of several key variables (Bene\_ID, Entry\_Date, Reentry\_Date, Discharge\_Date, Assessment\_Date, and MDS\_Version) and apply a single algorithm to group the assessments into stays with discrete entry and discharge dates. It is important to know that the file is at the assessment level, and in the MDS2 the entry date does not always show up on the first assessment in a stay (sometimes it doesn’t show up at all on a stay).

To begin, I sort the dataset by Bene\_ID and Assessment\_Date so that, theoretically, we can see a beneficiary’s chronological progression through various nursing homes. I start with the first observation we have for beneficiary and designate this as the starting point for the first stay. I then move on to the next observation. If there is a non-missing Discharge\_Date, I flag this as the last observation in the stay. In this case, the next observation for the same beneficiary would then represent the first assessment of the second stay. If the Discharge\_Date is missing I consider this an intermediate assessment within the first stay. I continue with this until I either find a discharge date or until a new beneficiary appears. Once I have flagged all of the assessments that I believe make up a single stay, I attempt to establish a definitive entry date and discharge date given the information on the assessments.

For the MDS2 I combine the Reentry\_Date and Entry\_Date, keeping the larger of the two in case they are both present on the same assessment. This is because if both the Reentry\_Date and Entry\_Date are provided, the Entry\_Date is likely for a preceding stay that should have been discharged before the reentry occurred. This phenomenon only occurs in the MDS2, and from here on Entry\_Date will refer to the union of Entry\_Date and Reentry\_Date. As a default, I set the entry date equal to the Entry\_Date on the first assessment for the stay. If Entry\_Date is missing, I use Assessment\_Date from the first observation. From there, I search all Entry\_Dates on all assessments in the stay. If one or more of the assessments indicates MDS\_Version==3, I use the Entry\_Date associated with these assessments because the MDS3 has significantly better Entry\_Dates that I trust more than those in the MDS2. If all assessments are MDS\_Version==2, then I take the maximum non-missing Entry\_Date that is less than the Discharge\_Date and impute this date as the official entry date on the stay. The reason I take the maximum Entry\_Date is because there are some cases in which the Entry\_Date field contains a value that is from a different admission in the past. This could either be a data entry error, or is some cases when the MDS stay is a readmission, the Entry\_Date field could contain the value of the original admission, which would be incorrect.

By design, there is only 1 Discharge\_Date per stay, or there are no Discharge\_Dates in the stay. In the event that there are no Discharge\_Dates, this implies that a beneficiary only has a single stay and none of the assessments associated with that beneficiary have a Discharge\_Date. I am currently interpreting this by assuming that the beneficiary is still in the nursing home as of the last day of our data. However, instead of using 12/31/2013 as the temporary discharge date, I use the date 1/1/2014 to denote that the beneficiary did not have an observable discharge during our observation window. There are two alternative ways to address this. We can either use the Assessment\_Date on the final observation we see for the beneficiary as a discharge date, or we can impute a discharge date 92 days after the last observed Assessment\_Date, assuming this occurs before the last observable date of our data. The rationale for this is that we would expect to see a quarterly assessment every 90 days, so if the beneficiary was in the nursing home for greater than 92 days, we would see another assessment, which does not appear.

**MDS Algorithm Version 2:** In version 2, I use a different method to create stays for the MDS2 and MDS3, and then append them together after collapsing down to the stay level individually, and attempt to smooth them together in the 4th quarter of 2010 when the transition occurs.

MDS2: The MDS2 construction process begins similarly to version 1. I sort by Bene\_ID, then chronologically by Assessment\_Date. I group assessments together based on what is between the first assessment and the first discharge. The assessment following that discharge is the first assessment in the 2nd stay. From here, I add nuance to the algorithm to try to better identify stays. Using an additional variable coding the primary reason for assessment (Assessment\_Reason) I can see if an individual assessment was an admission assessment, reentry assessment, discharge assessment, or other assessment. Theoretically, the first assessment in a stay should be either an admission assessment or reentry assessment, and the final one should be a discharge assessment. In practice, it is possible that an admission assessment happens in the middle of a stay where a discharge simply was not recorded. In these situations, it seems accurate to impute a Discharge\_Date before the intermediate admission assessment, and impute an Entry\_Date on the date of the admission assessment. The current rules for this are as follow:

* If an admission assessment or reentry assessment occurs in the middle of a stay, flag that stay.
* If a Discharge\_Date is missing and it is the last assessment for that beneficiary then flag that stay.
* If Assessment\_Reason indicates a discharge assessment (values 06, 07, or 08) then impute a discharge equal to the Assessment\_Date on that observation.
* If the Assessment\_Reason does not indicate a discharge, but we have flagged the observation for an imputed Discharge\_Date, then set the Discharge\_Date equal to the next observation’s Assessment\_Date (which should be for an admission assessment) IF the next observation has the same Bene\_ID AND the next observation’s Assessment\_Date comes within 90 days of the current Assessment\_Date.
* If the Assessment\_Reason does not indicate a discharge, but we have flagged the observation for an imputed Discharge\_Date, then set the Discharge\_Date equal midway between the current observation’s Assessment\_Date and the next observation’s Assessment\_Date (which should be for an admission assessment) IF the next observation has the same bene\_id AND the next observation’s Assessment\_Date comes later than 90 days after the current Assessment\_Date.
* If the Assessment\_Reason does not indicate a discharge, but we have flagged the observation for an imputed Discharge\_Date, then set the Discharge\_Date equal to either the Assessment\_Date+45 or September 30, 2010 (the last date observable in the MDS2) – whichever one is sooner – IF there are no more observations for this beneficiary. The rationale here is that it is likely that the beneficiary either continues to receive assessments in the MDS 3, or there is no discharge date due to administrative negligence so we impute a date halfway between when we see an assessment and when we would expect to see another quarterly assessment (90 days later).

After the new discharge dates have been imputed, we simply rerun our previous algorithm, which combs through assessments chronologically until it finds a discharge, then it groups all assessments between the first chronological occurrence and a discharge as part of the same stay. Using this method of imputing discharge dates according to assessment type, we gain about 120,000 stays per year out of about 3,600,000 unaltered stays per year – an increase of about 3.3%.

MDS3: The new algorithm for the MDS3 also relies on intermediate entry dates to impute discharge dates, though since the MDS3 has a more reliable and fully populated Entry\_Date variable, I simply use this variable instead of making assumptions based on Assessment\_Reason. To begin, I sort by bene\_id and Assessment\_Date. I take the first observation with a given Entry\_Date, and then I identify the last observation with the same Entry\_Date, and I group all intermediate assessments as being part of the same stay. In most cases there is a Discharge\_Date on the last observation in the stay. In cases where there is not, I apply the following decision rules:

* Set Discharge\_Date equal to Assessment\_Date + 45 if there is another entry for the same beneficiary that comes greater than 90 days in the future. The rationale here is that the beneficiary was released at some point before the next entry, but that the discharge was not recorded. I take the value that is halfway between the most recently observed Assessment\_Date and when I would have expected to see the next quarterly assessment had the beneficiary been in the nursing home continuously – which is 90 days away.
* Set the Discharge\_Date equal to the next Entry\_Date if the next entry is for the same beneficiary and the entry date is within 90 days of the current observation’s Assessment\_Date.
* Set the Discharge\_Date equal to December 31, 2100 in the event that it is the last observation for a beneficiary and there is still no Discharge\_Date.

To smooth the MDS2.0 and MDS3.0 together I begin by appending the two stay-level files together and sort by Bene\_ID and Entry\_Date. Presumably, if the MDS2.0 stay flows into the MDS3.0 file, we should see one stay with an Entry before or on 9/30/2010 with a discharge (which was imputed) at 9/30/2010. We should then see a stay in the next observation with an Entry\_Date occurring before 9/30/2010 and with a discharge after 9/30/2010. If these conditions are met, I keep the earliest Entry\_Date between the two adjacent stays, and the latest Discharge\_Date between the two adjacent stays. We now have two identical stays by Bene\_ID, Entry\_Date, and Discharge\_Date, so I drop stays that are duplicated on these variables.

**MDS Algorithm Version 3 (RHF):** This 3rd version is based on my interpretation of the Residential History File Algorithm described by Orna Intrator, Vince Mor, et al. (<https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3015013/>). This version differs from the other two because it does not create discrete MDS stays that try to mimic the other PAC stay level files, but rather uses characteristics of the assessment level MDS claims to backfill days in a nursing home. Thus, this version will not output a standalone MDS file, but rather will tack MDS days onto an existing file that is already organized at the episode level and assigns locations for day1-dayX. The general principle is as follows: if we see that an assessment took place in the raw MDS data, we can infer from the type of assessment (admission, quarterly, or annual) the upper bounds of how long the beneficiary had to be in the nursing home in order to receive this assessment. We then backfill that many days from the date of the assessment as days in a nursing home. Other PAC use (which has already been determined in previous steps) is used to fine-tune the guesswork in this process by either stopping the backfilling process or preventing a backfilling for the days where other PAC use is indicated. It is currently unclear from the RHF paper if the presence of other PAC use will completely stop the backfilling process for that assessment, or if it will just cause the algorithm to skip over those ‘occupied’ days and resume backfilling before the start of the other PAC use. Currently, I am stopping all backfilling when confronted with conflicting PAC care, based on this description in the RHF paper’s methods section:

*Once the calendar is populated by all information obtained from claims, remaining nonfilled days may be populated by projected NH days based on MDS assessment dates and type. For example, admission assessments are required within 2 weeks of admission; therefore, the RHF fills up to 14 days back during consecutive “gap” days to form an NH stay.*

The rules I am currently using to backfill days based on assessment type are as follows:

1. Admission Assessment goes back 14 days OR back to the most recent previous assessment for that beneficiary, whichever is most recent
2. Quarterly Assessment goes back 92 days OR back to the most recent previous assessment for that beneficiary, whichever is most recent
3. Annual Assessment goes back 365 days OR back to the most recent previous assessment for that beneficiary, whichever is most recent
4. Discharges go back 92 days OR back to the most recent previous assessment for that beneficiary, whichever is most recent. This is because we would expect to see an assessment at least once per quarter, so if the most recent assessment going back in time is >92 days before discharge then it is likely that there are either missing assessments or we have an erroneous discharge date, in which case we would not want to potentially impute >92 days of a nursing home stay without cause
5. There are Medicare PPS 5, 14, 30, 60, and 90 day assessments. Backfill the respective number of days OR back to the most recent previous assessment for that beneficiary, whichever is most recent
6. If any other, non-discharge assessment is present, fill in the day of the assessment as an MDS stay if the day has no other PAC stay affiliated with it, but make no further assumptions about backfilling.