In large part, the following describes what happens in /proj/DaltonLab/projects/p0013/progs/02\_CCF\_cohort\_data.Rmd

Essentially, this document creates views and tables that lightly clean the raw tables of CCF data and pare them down to only include patients in the NEOCARE cohort (i.e., they are inner-joined with DL\_NEOCARE.STUDY\_ID\_KEY). This relies on the QHS\_OUTCOMES\_V database (call Alex Milinovich).

Code chunk “demogs”

* Creates DL\_NEOCARE.CCF\_COHORT\_DEMOGRAPHICS\_V on Teradata. In essence, this is DL\_NEOCARE.STUDY\_ID\_KEY inner joined with QHS\_OUTCOMES\_V.Patients on PatientID left joined with DL\_NEOCARE.ODI\_CauseOfDeath on PatientID
* Columns
  + study\_id (integer)
  + birth\_date (date)
  + female (byteint)
    - 1 if QHS\_OUTCOMES\_V.Patients.Gender was C0086287 (“Females”)
    - 0 if QHS\_OUTCOMES\_V.Patients.Gender was C0086582 (“Males”)
    - NULL otherwise (there were no other values in QHS\_OUTCOMES\_V.Patients.Gender but some were missing)
  + hispanic\_or\_latino (byteint)
    - In order:
      * If QHS\_OUTCOMES\_V.Patients.RACE was C0019576 (“Hispanic Americans”)…
        + 0 if QHS\_OUTCOMES\_V.Patients.ETHNICITY was “Not Hispanic”
        + 1 otherwise
      * 1 if QHS\_OUTCOMES\_V.Patients.ETHNICITY was C0019576 (“Hispanic Americans”)
      * 0 if QHS\_OUTCOMES\_V.Patients.ETHNICITY was C1518424 (“Not Hispanic or Latino”)
      * NULL otherwise
  + race (varchar 38)
    - In order:
      * “White” if QHS\_OUTCOMES\_V.Patients.RACE or QHS\_OUTCOMES\_V.Patients.ETHNICITY was C0007457 (“Caucasoid race”)
      * “Black/African American” if QHS\_OUTCOMES\_V.Patients.RACE was C0005680 (“Black race”)
      * “Black/African American” if QHS\_OUTCOMES\_V.Patients.ETHNICITY was C0085756 (“African American”)
      * “Asian” if QHS\_OUTCOMES\_V.Patients.RACE was C0078988 (“Asians”)
      * “American Indian/Alaska Native” if QHS\_OUTCOMES\_V.Patients.RACE was C1515945 (“American Indian or Alaska Native”)
      * “Native Hawaiian/Other Pacific Islander” if QHS\_OUTCOMES\_V.Patients.RACE was C1513907 (“Native Hawaiian or Other Pacific Islander”) or C1531604 (“Asian or Pacific Islander”)
      * “Multiracial” if QHS\_OUTCOMES\_V.Patients.RACE was C1881928 (“Multiracial”)
      * NULL otherwise
  + death\_date (date)
    - The first nonmissing among:
      * QHS\_OUTCOMES\_V.Patients.OhioDeathIndexDate,
      * QHS\_OUTCOMES\_V.Patients.SocialSecurityDeathDate
      * QHS\_OUTCOMES\_V.Patients.DateOfDeath
      * DL\_NEOCARE.ODI\_CauseOfDeath.OhioDeathIndexDate
    - Coerced to missing, however, if outside the range 1999-2017
  + icd10cod (varchar 5)
    - coerced to missing if death\_date (above) was missing
  + cv\_death (byteint)
    - coerced to missing if death\_date (above) was missing
  + EntityAxisCodes (varchar 160)
    - coerced to missing if death\_date (above) was missing
* Notes:
  + The table DL\_NEOCARE.ODI\_CauseOfDeath was used to supplement the sometimes incomplete death data in QHS\_OUTCOMES\_V.Patients.

Code chunk “encs”

* Creates DL\_NEOCARE.CCF\_COHORT\_ENCOUNTERS\_V on Teradata. In essence, this is DL\_NEOCARE.STUDY\_ID\_KEY inner joined with QHS\_OUTCOMES\_V.Encounters on PatientID.
* Columns:
  + study\_id (integer)
  + CONTACT\_DATE (timestamp 3)
  + height\_cm (number (40, 0))
    - converted from text strings of the form e.g. 5' 4"
  + weight\_kg (float)
    - converted from weight in ounces
  + bmi (decimal (9, 2))
  + sbp (float)
  + dbp (float)
* Inclusion
  + Encounters included if AppointmentStatus was C1548411 (“Completed”) or C1706079 (“Arrived”) and if CancelationReason was NULL.
  + Excluded if not during 1999-2017

Section “Height, Weight, and Blood Pressure”

* Left unfinished.
* The idea is to incorporate height, weight, and blood pressure from QHS\_OUTCOMES\_V.Flowsheets.
* For even more height/weight/bp data, develop this.

Diagnoses section

* The code chunk icd\_range\_to\_single\_keys produces SQL code used in the diags code chunk (see below. It uses QHS\_UMLS\_V.Atoms to find the most preferable *single-code* mapping from ConceptID to ICD code when the most preferable mapping is a hyphenated range of codes.
* Code chunk “diags”.
  + Creates DL\_NEOCARE.CCF\_COHORT\_DIAGNOSES\_V. In essence, this is DL\_NEOCARE.STUDY\_ID\_KEY inner joined with QHS\_OUTCOMES\_V.Diagnoses on PatientID.
  + Columns
    - study\_id (integer)
    - diag\_date (timestamp 3)
    - ConceptID (character 8)
    - icd9 (varchar 100)
      * When original QHS\_OUTCOMES\_V.ICD9Code was a range, it was recoded to a corresponding single code when possible, based on the value of ConceptID
      * Therefore, be aware that there may be hyphenated ranges
    - icd10 (varchar 100)
      * When original QHS\_OUTCOMES\_V.ICD10Code was a range, it was recoded to a corresponding single code when possible, based on the value of ConceptID
  + Diagnoses excluded if not in the range 1999-2017. Diagnosis\_date used unless missing, and in that case contact date was used.
  + Where possible, ICD codes that were coded as ranges were recoded to corresponding single codes. This was accomplished using the code generated by the code chunk icd\_range\_to\_single\_keys (see above)
* Elixhauser subsection
  + Creates DL\_NEOCARE.CCF\_COHORT\_ELIX\_V, a shortcut to all Elixhauser diagnoses (based on both ICD9 and ICD10) found in CCF\_COHORT\_DIAGNOSES\_V.
  + Columns:
    - study\_id (integer)
    - diag\_date (timestamp 3)
    - elix (varchar 12)
  + joins DL\_NEOCARE.CCF\_COHORT\_DIAGNOSES\_V (see above) to the icd-to-elixhauser key tables created in /proj/DaltonLab/projects/neocare/p0013/progs/icd\_elix\_key\_table\_creation.R
  + Converts all ICD9 codes to Elixhauser, then converts all ICD10 codes to Elixhauser, then performs UNION, keeping unique combinations
* CCS subsection
  + Creates DL\_NEOCARE.CCF\_COHORT\_CCS\_DIAGNOSES\_V
  + Columns
    - study\_id (integer)
    - diag\_date (timestamp 3)
    - ccs (integer)
  + joins DL\_NEOCARE.CCF\_COHORT\_DIAGNOSES\_V (see above) to the icd-to-ccs key tables created in /proj/DaltonLab/projects/neocare/p0013/progs/icd\_ccs\_key\_table\_creation.R
  + Converts all ICD9 codes to CCS, then converts all ICD10 codes to CCS, then performs UNION, keeping unique combinations

Procedures section

* CPT-to-CCS crosswalk downloaded from <https://www.hcup-us.ahrq.gov/toolssoftware/ccs_svcsproc/ccscpt_downloading.jsp> and uploaded to Teradata as DL\_NEOCARE.procedure\_cpt\_to\_ccs\_key. This was joined with each source table so that CCS classifications would be at hand.
* Creates DL\_NEOCARE.CCF\_COHORT\_PROCEDURES\_V.
  + In essence, this is QHS\_OUTCOMES\_V.Procedures UNIONed with QHS\_OUTCOMES\_V.Surgeries and inner joined with DL\_NEOCARE.STUDY\_ID\_KEY, then full outer joined with QHS\_OUTCOMES\_V.SurgicalHistory. The latter’s data never superseded any of the former’s data; it only acted as a supplement.
  + Columns:
    - study\_id (integer)
    - proc\_date (timestamp 3)
    - ConceptID (character 8)
    - cpt (varchar 100)
    - CCS (smallint)
  + Included if proc\_date was in 1999-2017
* The section “Non CPT match analysis” is an exploration of ways to categorize procedures with CCS codes when they don’t have CPT codes. Largely unsuccessful.

Medication section

* Creates DL\_NEOCARE.CCF\_COHORT\_MEDICATIONS\_V.
* QHS\_OUTCOMES\_V.Medications inner joined with DL\_NEOCARE.STUDY\_ID\_KEY
* Columns:
  + study\_id (integer)
  + ConceptID (character 8)
  + dosage (varchar 255)
  + doseunit (character 8)
  + doseunitdescription (varchar 900)
  + OrderDate (timestamp 3)
  + StartDate (timestamp 3)
  + EndDate (timestamp 3)
* Rows included if their order date, start date, or end date was between 1999 and 2017, their ConceptID was not NULL, and their OrderingMode was C1547560 (“Outpatient”)
* Meds of Interest section contains the code chunk that loads the fruits of the labor of /proj/DaltonLab/projects/neocare/p0013/progs/med\_work.Rmd
  + In order to find all relevant ConceptIDs for a medication of interest, its children are collected using QHS\_UMLS\_V.Relationships where ExpandedRelationship = “isa”.
  + Doug Einstadter manually selected the child concepts that can be appropriately deemed a subset or member of the medication of interest. Each of these child concepts was fed to the MedicationConceptsTesting stored procedure (found in the “Medication Concepts” section of /proj/DaltonLab/projects/neocare/p0013/progs/stored\_procedures.Rmd) in order to grab all appropriate ConceptIDs
  + This was done for antihypertensives, antidiabetes medication, and antidepressants
* In general, in order to isolate a desired group of medications, find an overarching ConceptID that encompasses the desired medications with satisfactory specificity, and feed that to the MedicationConceptsTesting stored procedure. Manually review the results.

Labs section

* Lab components are classified with a granularity finer than what we were interested in researching, so we manually tagged the granular lab components such that they would be members of human-readable lab result groups of interest (e.g., HDL cholesterol, RDW, glucose).
* Distributions and units of each granular lab component were examined to see which required a conversion factor in order to standardize result values. Lab components with relatively tiny percentage within their lab groups of interest were dropped.
* The subsection “More Investigation” took a lot of work in order to understand how to perform the things described above so I retained it, but it’s largely not needed at this point.
* The Results section contains views individualized to grab only certain lab results of interest.
  + Columns are study\_id (integer), ResultDate (timestamp 3), and the third column is named after the result.
* In general, in order to isolate a new desired group of lab results, follow the instructions atop the “ComponentID to Component” section
  + The code in the section yields a table (“ccf\_lab\_component\_unit\_summary.csv”) that contains the counts and distributions of each combination of ComponentID and Unit.
  + Using the human-readable columns Component and UnitDescription, the researcher can select the rows that seem to contain the labs of interest. The user then writes a query that grabs only lab values with those ComponentID/Unit combinations that the user identified. The user may have to convert some combinations’ values so that all values are using the same units.
  + Everything within the “ComponentID to Component” section that follows the aforementioned table creation is a demonstration of how we employed this process. I recommend, however, not getting bogged down with the details of this demonstration because it contains misleading redundancy as a result of teaching ourselves how to do this.
* The “More investigation” section contains summaries that were written as we sought to understand what QHS\_OUTCOMES\_V.Results contained.
* The “Results” section contains the view-creating queries that grab specific labs of interest. Use these as models when creating your own. Notice how they are simply pulls of QHS\_OUTCOMES\_V.Results with IN lists.

GEOIDs section

* First, city\_zcta\_key was created.
  + The shapefiles of each ZCTA and each city was intersected. A row was kept only if there was any intersection, and the area was kept as a column.
  + Columns:
    - city\_geoid
    - city\_name
    - zcta
    - area – the area of the intersection of the city and zcta shapefile
    - city population
    - state abbreviation
    - county geoid – this works because we used shapefiles that split trans-county cities
    - preference
      * an order of preference when all other tiebreaking procedures (see below) do not break a tie
      * "39035" = 1, "39093" = 2, "39103" = 3, "39153" = 4, "39055" = 5, "39085" = 6, "39133" = 7, all other Ohio = 8, outside of Ohio = 9
* Creates DL\_NEOCARE.CCF\_COHORT\_GEOIDS
* DL\_NEOCARE.CCF\_PatientAddressesGeocoded INNER JOINED WITH DL\_NEOCARE.STUDY\_ID\_KEY.
* Rows with no GEOID were thrown out unless IsHomeless = 1. In attempt to locate such patients, the GEOID of the patient’s city was plugged in where possible. The row’s city, state, and zip were left joined to city\_zcta\_key (see above) in several different ways in attempt determine the city (using city\_zcta\_key.city\_geoid):
  + Join by city, zip, and state. If multiple matches, select the row with the largest city/zcta intersectional area, then the largest city population, then the smallest “preference”, then the lowest city GEOID. If no matches, go to next join:
  + Join by city and zip. If multiple matches, select the row with the largest city/zcta intersectional area, then the largest city population, then the smallest “preference”, then the lowest city GEOID. If no matches, go to next join:
  + Join by city and state. If multiple matches, select smallest “preference”, then largest population, then lowest city GEOID. If no matches, go to next join:
  + Join by zip and state. If multiple matches, select smallest preference, then smallest city/zcta intersectional area, then largest population, then smallest city GEOID. If no matches, go to next join:
  + Join by city name only. If multiple matches, select smallest preference, then largest population, then smallest city GEOID. If no matches, go to next join:
  + Join by zip only. If multiple matches, select largest area, then largest city population, then smallest “preference”, then smallest city GEOID. If no matches, give up.
* Columns:
  + study\_id (integer)
  + geoid\_date (date)
  + geoid (varchar 12),
  + homeless (byteint)
* There were many cases where the raw data associated multiple different GEOIDs with the same patient on the same date. An algorithm disambiguated each instance as follows:
  + GEOIDs were shortened to only include their first 12 characters, effectively making them into block group GEOIDs instead of block GEOIDs. GEOIDs flagged as homeless were replaced with “homeless”.
  + GEOIDs’ end dates were filled with the date 9999-12-31 if missing.
  + Rows were excluded GEOID or its start date was missing or if its start date was later than its end date
  + The data were aggregated such that each unique combination of study\_id, GEOID, and GEOID start date was kept, along with the combination’s latest occurring value of EFF\_END\_DATE.
  + For each unique combination of study\_id and GEOID start date in the data:
    - If the combination only appeared in one row, that row was kept.
    - If the combination appeared in more than one row, then any of these rows that had EFF\_START\_DATE = EFF\_END\_DATE were tossed. Exception: If all these rows had EFF\_START\_DATE = EFF\_END\_DATE, they were all kept. (The idea here is to presume that GEOIDs that began and ended on the same day were entered by mistake and corrected to the value that had a later EFF\_END\_DATE.)
  + Two iterative processes were applied in which patient/date combinations with multiple GEOIDs were disambiguated as much as possible.
    - Essentially, if one of the GEOIDs was the same as the directly earlier GEOID, it was removed. Then, if one of the GEOIDs was the same as the directly later GEOID, all others were removed.
    - In excruciating detail:
      1. Duplicate rows were removed.
      2. An iterative process in which rows were removed was applied to the table over and over again until no more rows met the criteria for removal. The table was sorted by study\_id followed by start\_date, and then any given row (call it Row X) was removed if these three conditions were met:
         * Row X’s combination of study\_id and start\_date was not unique in the table (i.e., at least one other row had the same study\_id and start\_date as Row X).
         * Considering only rows with the same study\_id as Row X, there existed at least one start\_date earlier than Row X’s start\_date, and the largest of these start\_dates occurred in only one row, which we will call Row Y (i.e., the patient’s next earliest start\_date has a single GEOID associated with it).
         * Row X and Row Y have the same GEOID.
      3. An iterative process in which rows were removed was applied to the table over and over again until no more rows met the criteria for removal. The table was sorted by study\_id followed by start\_date, and then any given row (call it Row J) was removed if these three conditions are met:
         * Row J’s combination of study\_id and start\_date is not unique in the table (i.e., at least one other row had the same study\_id and start\_date as Row J).
         * Considering only rows with the same study\_id as Row J, there existed at least one start\_date earlier than Row J’s start\_date, and the largest of these start\_dates occurred in only one row, which we will call Row K (i.e., the patient’s next earliest start\_date has a single GEOID associated with it).
         * Row J and Row K have the same GEOID.

Homelessness section

* Inner join of DL\_NEOCARE.CCF\_PatientAddressesGeocoded and DL\_NEOCARE.STUDY\_ID\_KEY
* Includes only the unique dates during 1999-2017 wherein each patient was flagged as homeless.
* Columns:
  + study\_id (integer)
  + homelessness\_date (date)

Financial class section

* Financial class is on a per-encounter basis in order to try to get financial class per the same patient at different times.
* Encounters can merge with various financial tables. So, we pulled all combinations of five variables from three different financial tables and did an analysis to see the best way to categorize encounters.
* The five variables were payor name, financial class of payor, plan name, product type of plan, and financial class of coverage.
* Finally, a view is created once the categorization procedure was developed.
  + Called DL\_NEOCARE.CCF\_COHORT\_FINANCIAL\_CLASS\_V
  + STUDY\_ID\_KEY inner joined with QHS\_OUTCOMES\_V.Encounters left joined with IHAA\_EDV.BK\_PAT\_ENC left joined with IHAA\_EDV.HR\_ENC\_PRIMARY left joined with IHAA\_EDV.BK\_PAYOR left joined with IHAA\_EDV.BK\_PLAN left joined with IHAA\_EDV.BK\_CVG
  + Columns:
    - study\_id (integer)
    - CONTACT\_DATE (date)
    - financial\_class (varchar 36)
  + When multiple financial classes are associated with a study\_id / CONTACT\_DATE combination, one is chosen according to the following order of precedence:
    - Military
    - Medicaid
    - Medicare
    - Private
    - Research/Worker’s Comp/Other/Unknown
    - Uninsured/Self-pay

Smoking status section

* Smoking status is on a per-encounter basis in order to try to get smoking status per the same patient at different times.
* View called DL\_NEOCARE.CCF\_COHORT\_SMOKING\_STATUS\_V
  + STUDY\_ID\_KEY inner joined with QHS\_OUTCOMES\_V.SocialHistory
  + Many patient/ContactDate combinations were represented by multiple rows, some having conflicting data. These multiple rows were distilled down to one row, using logic described below to select one of the conflicting values in the data.
  + Columns:
    - study\_id (integer)
    - contact\_date (date)
    - smoking\_status (varchar 7)
      * A two-step process was employed to select among conflicting values within patient/ContactDate combinations:
        1. The “maximum” extant QHS\_OUTCOMES\_V.TobaccoStatus value was selected, where ‘current’ > ‘former’ > ‘passive’ > ‘never’ > missing

Exception: if the maximum extant value was ‘passive’, ‘never’ or missing and QHS\_OUTCOMES\_V.SmokingQuitDate was not missing, ‘former’ was assigned

* + - * 1. After step 1 was completed, if smoking\_status was ‘former’ on at least two preceding dates, values of ‘passive’, ‘never’ and missing were changed to ‘former’.
    - quit\_date (date)
      * latest value of QHS\_OUTCOMES\_V.SmokingQuitDate per patient/ContactDate combination
    - packs\_per\_day (integer)
      * Largest value of QHS\_OUTCOMES\_V.PacksPerDay per patient/ContactDate combination
    - years\_smoked (integer)
      * Largest value of QHS\_OUTCOMES\_V.YearsUsed per patient/ContactDate combination