**Readme file for programming code**

These notes describe the procedure for using the code files when one conducts the analysis. Explanations for specific decisions, such as the diagnosis codes used to identify trauma-related injuries, are provided elsewhere in the main manuscript and supplementary materials. All code files listed below are available at <https://github.com/sanghavi-lab/tcp_regression_paper>.

**Software**

We used python 3.8.2, R 3.6.1, and Stata/MP 17.0 for this analysis.

**Step 1: Gather and construct relevant datasets from Medicare**

The first file (01\_export\_to\_parquet) creates the files needed to construct the analytical sample, by year. The second file (02\_obtain\_mileage\_information) links the mileage information with the ambulance claims created from 01\_export\_to\_parquet.

|  |  |  |  |
| --- | --- | --- | --- |
| Script name | Description | Input files | Output files |
| 01\_export\_to\_parquet (python) | Construct inpatient, outpatient, and ambulance claims. | 1. MedPAR  2. Carrier base file  3. Carrier line file  4. Master Beneficiary Summary File  5. Outpatient base file | 1. Inpatient claims  2,3,4. Ambulance claims  5. Outpatient claimsa |
| 02\_obtain\_mileage\_information (python) | Obtain mileage information for each ambulance claim. | 1. Carrier line file  2. Ambulance claims | 1. Ambulance claims with mileage information |

Notes: The numbers indicate which input files were used to create the output files. (e.g. the carrier base (2), line (3), and master beneficiary summary (4) files were used to create the ambulance claims (2,3,4))

aBefore merging the raw institutional claims with the ambulance claim in step 2, we created a subset of the outpatient file consisting of only op claims that matched the with the ambulance claims. This is to ensure efficient processing since the op file was large.

**Step 2: Identify beneficiaries who went to the hospital with an emergency ambulance ride**

The python script below merges the institutional claims (inpatient and outpatient claims) with the ambulance claims to obtain information on beneficiaries who took an emergency ambulance ride, by year. We keep all institutional claims and create an indicator if the beneficiary took an emergency ambulance ride to the hospital.

|  |  |  |  |
| --- | --- | --- | --- |
| Script name | Description | Input files | Output files |
| 03\_identify\_institutional\_claims\_with\_amb\_ride (python) | Gather ambulance information for any beneficiary who took an ambulance ride. We also ensured that all of the outpatient claims were from the hospital (CLM\_FAC\_TYPE\_CD = 1) | 1. Ambulance claims with mileage information  2. Inpatient claims  3. Outpatient claims | 1,2. Inpatient claims with ambulance information  1,3. Outpatient claims with ambulance information |

Notes: The numbers indicate which input files were used to create the output files. (e.g. the ambulance (1) and inpatient (2) claims were used to create the inpatient claims with ambulance information (1,2))

**Step 3: Reduce institutional file to keep relevant claims for the analysis**

There are five total python scripts (see below) that aim to reduce the institutional file by dropping non-trauma related, rural, or duplicated claims or claims as a result of a hospital-to-hospital transfer, by year. The last script (08\_clean\_and\_concat) combines the inpatient and outpatient files into one file.

|  |  |  |  |
| --- | --- | --- | --- |
| Script name | Description | Input files | Output files |
| 04\_identify\_trauma (python) | Keep beneficiaries with a valid injury code.a | 1. Inpatient claims with ambulance information  2. Outpatient claims with ambulance information | 1. Inpatient claims with at least one valid injury code (ICD-9CM or ICD-10CM)  2. Outpatient claims with at least one valid injury code (ICD-9CM or ICD-10CM) |
| 05\_Identify\_FFS\_and\_nonrural (python) | Create an indicator for fee-for-service claims (Medicare parts A and B) and keep claims from nonrural areas. | 1. Inpatient claims with at least one valid injury code (ICD-9CM or ICD-10CM)  2. Outpatient claims with at least one valid injury code (ICD-9CM or ICD-10CM)  3. Master Beneficiary Summary File  4. Federal Office of Rural Health Policy data to identify rural areas | 1,3,4. Inpatient claims from nonrural areas  2,3,4. Outpatient claims from nonrural areas |
| 06\_drop\_op\_ip\_duplicates (python) | Drop any duplicates if two or more claims have the same beneficiary identification (BENE\_ID) and service start date (ADMSN\_DT or CLM\_FROM\_DT) | 1. Inpatient claims from nonrural areas  2. Outpatient claims from nonrural areas | 1. Inpatient claims without duplicated BENE\_ID and ADMSN\_DT (admission date)  2. Outpatient claims without duplicated BENE\_ID and CLM\_FROM\_DT (claim from date) |
| 07\_identify\_first\_stops (python) | Use the raw institutional claims (i.e. all inpatient and outpatient claims) to help identify and keep institutional claims from the analytical sample containing only hospitals considered the first stop after an injury event (i.e. dropped any claim as a result of a transfer from another hospital). | 1. Raw inpatient file  2. Raw outpatient file  3. Analytical inpatient claims without duplicated BENE\_ID and ADMSN\_DT (admission date)  4. Analytical outpatient claims without duplicated BENE\_ID and CLM\_FROM\_DT (claim from date) | 1,2,3. Inpatient claims containing only first destination hospitals  1,2,4. Outpatient claims containing only first destination hospitals |
| 08\_clean\_and\_concat (python) | Concatenate the inpatient claims with the outpatient claim into one file. We created a couple of indicators (e.g. dual eligibility status) | 1. Inpatient claims containing only first destination hospitals  2. Outpatient claims containing only first destination hospitals  3. Master Beneficiary Summary File | 1,2,3. Concatenated institutional claims (analytical sample) |

Notes: The numbers indicate which input files were used to create the output files. (e.g. the inpatient claims file containing ambulance information (1) was used to create the inpatient claims file containing at least one valid injury ICD-9CM code (1))

aValid injury codes (ICD-9CM or ICD-10CM) are defined by the Healthcare Cost and Utilization Project (HCUP) (see supplement)

**Step 4: Prepare files for ICPIC-R software to calculate the new injury severity scores**

The following scripts will prepare files for R, calculate the various measures of injury severity (e.g. new injury severity score and abbreviated injury scale) using the ICDPIC-R package in R (<https://injepijournal.biomedcentral.com/articles/10.1186/s40621-018-0149-8>), and link the severity information with the analytical file.

|  |  |  |  |
| --- | --- | --- | --- |
| Script name | Description | Input files | Output files |
| 09\_prepare\_claims\_for\_icdpicr (python) | Prepare the claims file for ICDPIC-R by dropping duplicated diagnosis codes. | 1. Concatenated institutional claims (analytical sample) | 1. ICD-9CM institutional claims containing only an identifier and diagnosis columns (2011-2015)  1. ICD-10CM institutional claims containing only an identifier and diagnosis columns (2016-2017) |
| 09a\_niss\_calculations\_in\_R [folder] (R) | This folder contains two R scripts: one to calculate injury severity from ICD-9CM codes and another to calculate injury severity from ICD-10CM codes using the ICDPIC-R package | 1. ICD-9CM institutional claims containing only an identifier and diagnosis columns (2011-2015)  2. ICD-10CM institutional claims containing only an identifier and diagnosis columns (2016-2017) | 1. ICD-9CM dataset containing severity information for each injury (2011-2015)  2. ICD-10CM dataset containing severity information for each injury (2016-2017) |
| 10\_obtain\_niss\_from\_icdpicr (python) | Merge output files from ICDPIC-R with the analytical file (concatenated institutional claims) to obtain injury severity information and keep only beneficiaries with major trauma. | 1. ICD-9CM dataset containing severity information for each injury (2011-2015)  2. ICD-10CM dataset containing severity information for each injury (2016-2017)  3. Concatenated institutional claims (analytical sample) | 1,2,3. Institutional claims with injury severity information for beneficiaries with major trauma (analytical sample) |

Notes: The numbers indicate which input files were used to create the output files. (e.g. the concatenated institutional claims file (1) was used to prepare two files for ICDPIC-R: ICD-9CM (1) and ICD-10CM (1) institutional claims)

**Step 5: Prepare files for SAS software to calculate the combined Charlson-Elixhauser comorbidity scores**

The following scripts prepare files for SAS by gathering diagnosis codes up to one year prior to the injury event, calculate the combined Charlson-Elixhauser comorbidity score for each beneficiary, and link the scores back with the analytical file. Additionally, the last script in this list will create some indicators/measures and remove any contiguous or overlapping institutional claims.

|  |  |  |  |
| --- | --- | --- | --- |
| Script name | Description | Input files | Output files |
| 11\_obtain\_dx\_from\_raw\_for\_comorbidity\_scores (python) | Gather diagnosis codes from raw inpatient, outpatient, and ambulance claims. Since the raw files are too large, the goal is to reduce these files by collecting only diagnosis codes for beneficiaries in the analytical sample. | 1. Inpatient claims  2. Outpatient claims  3. Ambulance claims  4. Analytical sample containing beneficiaries with major trauma | 1,2,3,4. File containing diagnosis information from inpatient, outpatient, and ambulance claims for only beneficiaries in the analytical sample |
| 12\_prep\_files\_for\_comorbidity\_calculations.py | Merge the reduced raw inpatient, outpatient, and ambulance claims from the previous script with the analytical sample and  keep diagnosis codes up to one year prior to the injury event. Create a long data format in preparation for SAS. | 1. Analytical sample containing beneficiaries with major trauma  2. File containing diagnosis information from inpatient, outpatient, and ambulance claims from the previous script | 1,2. Long data format of beneficiaries containing diagnosis codes from the year before the emergency event |
| 12a\_comorbidity\_scores\_calculations\_in\_SAS [folder] (SAS) | This folder contains one SAS script that creates a wide-data format and calculates the combined Charlson-Elixhauser comorbidity score for each beneficiary using diagnosis codes up to one year prior to the injury event. | 1. Long data format of beneficiaries containing diagnosis codes from the year before the emergency event | 1. Wide data format containing a comorbidity score for each beneficiary |
| 13\_obtain\_comorbid\_and\_other\_tasks (python) | First, merge output file from SAS with the analytical file to obtain comorbidity scores for each beneficiary. Second, create the following information:  - Indicators for the three mechanisms of injury for the subgroup analysis (falls, motor vehicle accidents, and firearm injuries)  - Indicators for mortality at 30, 90, 180, and 365 days  - Age based on the starting service date and birthdate  Third, remove remaining duplicates in the inpatient portion of the analytical file due to merging in previous scripts and drop any outpatient claims that are contiguous or overlapping. | 1. Wide data format containing a comorbidity score for each beneficiary  2. Analytical sample containing beneficiaries with major trauma | 1,2. Analytical sample without any contiguous or overlapping claims and with comorbidity, indicator, and age information |

Notes: The numbers indicate which input files were used to create the output files. (e.g. the inpatient (1), outpatient (2), ambulance (3) claims, and analytical sample containing beneficiaries with major trauma (4) were used to create the file containing diagnosis information from inpatient, outpatient, and ambulance claims for only beneficiaries in the analytical sample (1,2,3,4))

**Step 6: Create additional indicators/measures for the analytical sample**

The below scripts create the following information: indicators for beneficiaries with a transfer (i.e. beneficiaries with a first- and second-stop), pick-up/drop-off location indicators for those who took an ambulance ride, 29 chronic condition indicators, geographic measures for adjustments, and hospital characteristics (e.g. hospital volume and other proxy quality measures).

|  |  |  |  |
| --- | --- | --- | --- |
| Script name | Description | Input files | Output files |
| 14\_obtain\_raw\_for\_transfer\_identification (python) | Gather institutional claims from raw inpatient and outpatient files. These will be used to identify which beneficiaries from the analytical sample had a transfer to another hospital (a second-stop). Since the raw files are too large, the goal is to reduce these files by collecting only institutional claims relevant to beneficiaries in the analytical sample. | 1. Inpatient claims  2. Outpatient claims  3. Analytical sample without any contiguous or overlapping claims | 1,2,3. All institutional claims relevant for only beneficiaries in the analytical sample |
| 15\_identify\_transfers (python) | Merge the reduced raw institutional claims from the previous script with the analytical sample. Then, create an indicator to identify any beneficiary from the analytical sample that had a transfer to another hospital within two days of the discharge date. | 1. All institutional claims relevant for only beneficiaries in the analytical sample (from previous script)  2. Analytical sample without any contiguous or overlapping claims | 1,2. Analytical claims with transfer information |
| 16\_gather\_more\_measures\_for\_model (python) | Create the following:  - Pick-up/drop-off location indicators for those who took an ambulance ride  - 29 chronic condition indicators  - Geographic measures for adjustments  - Hospital characteristics (e.g. hospital volume and other proxy quality measures) | 1. Analytical claims with transfer information  2. Master Beneficiary Summary File with Chronic Conditions and Other Chronic Conditions information  3. Area Health Resource File (geographic controls)  4. Dartmouth Atlas files (number of hospital beds)  5. Medicare’s Hospital Compare data (other proxy quality measures) | 1,2,3,4,5. Analytical sample with various indicators/measures |

Notes: The numbers indicate which input files were used to create the output files. (e.g. the inpatient (1), outpatient (2), and analytical sample without any contiguous or overlapping claims (3) were used to create the file with all institutional claims relevant for only beneficiaries in the analytical sample (1,2,3))

**Step 7: Identify trauma and non-trauma centers in the analytical sample**

The following script use the American Trauma Society trauma registry to identify a trauma center as any hospital having a designated level. We defined non-trauma centers as any hospital without a designated level. Various factors must be considered to accurately identify a hospital type (trauma centers vs non-trauma centers). These factors were commented and justified throughout the script. The summary of the merging process is provided at the top of the python file.

|  |  |  |  |
| --- | --- | --- | --- |
| Script name | Description | Input files | Output files |
| 17\_identify\_hospital\_types (python) | Create two files:  - Hospitals with a trauma center. Removed any hospitals owning more than one trauma centers with varying levels (e.g. a hospital with both levels 1 and 2) and hospitals that changed levels between 13-17  - Hospitals without a trauma center | 1. American Trauma Society file  2. American Hospital Association crosswalk file  3. Analytical sample with various measures | 1,2,3. Final analytical file containing only hospitals with a trauma center  1,2,3. Final analytical file containing only hospitals without a trauma center |

Notes: The numbers indicate which input files were used to create the output files. (e.g. the American Trauma Society (1), American Hospital Association (2), and Analytical sample with various measures (3) files were used to construct both final analytical files containing hospitals with or without a trauma center (1,2,3))

**Step 8: Create risk-adjusted hospital surgical quality measures**

The scripts below will create the risk-adjusted surgical quality measures for each hospital. We did this by regressing 30-day mortality on age, sex, race, comorbidity scores, and fixed effects for surgical Diagnosis-Related Groups and each year, using all inpatient claims with a surgical DRG code between 2011-2017 (see “A9 Creating hospital surgical quality scores” to see which surgical DRG codes we used). We squared the age and comorbidity variables in the model. We removed all transfers from other hospitals, admissions from the Emergency Department, and beneficiaries with an injury code. Therefore, this sample did not overlap with our sample. Then, we subtracted the modeled mortality probability from the binary 30-day mortality indicator for each observation. Finally, we averaged these residuals for each hospital. Because we worked with 30-day mortality instead of survival, we negated the average residual for each hospital so that a more positive value is associated with higher quality of care before creating figure 2.

|  |  |  |  |
| --- | --- | --- | --- |
| Script name | Description | Input files | Output files |
| 18\_hospital\_quality\_measures [folder] (python, SAS, & Stata) | There are a total of seven scripts in this folder – 5 python, 1 SAS, and 1 Stata files. The following are descriptions for each script:  A. 01\_obtain\_bene\_w\_surgical\_drgs (python)  - Selects patients with surgical DRGS from the raw inpatient file.  B. 02\_gather\_dx\_from\_raw\_hos\_quality (python)  - Similar to step 5, this gathers diagnosis codes from raw inpatient, outpatient, and ambulance claims. Since the raw files are too large, the goal is to reduce these files by collecting only diagnosis codes for beneficiaries in the inpatient record with a surgical DRG code.  C. 03\_prep\_files\_to\_calc\_comorbidities (python)  - Similar to step 5, this merge the reduced raw inpatient, outpatient, and ambulance claims from the previous script with beneficiaries containing only surgical DRG codes and keep diagnosis codes up to one year prior to the admission date. Create a long data format in preparation for SAS.  D. comorbidity\_scores\_sas\_code\_hos\_quality (SAS)  - This file is within the folder labeled 03a\_comorbidity\_scores\_calculations\_in\_SAS. Similar to step 5, this creates a wide-data format and calculate the combined Charlson-Elixhauser comorbidity score for each beneficiary using diagnosis codes up to one year prior to the admission date.  E. 04\_merge\_comorbid\_w\_claims (python)  - First, merge output file from SAS with the inpatient file containing surgical DRG codes to obtain comorbidity scores for each beneficiary. Second, create the following information:  - Indicators for mortality at 30 days  - Age based on the starting admission date and birthdate  F. 05\_drop\_claims\_w\_trauma\_for\_stata (python)  - Drop all claims containing an injury code and create year fixed effects.  G. creating\_hos\_qual\_measures (Stata)  - This file is within the folder labeled 05a\_hos\_qual\_scores\_calculations\_in\_stata. This models mortality probabilities using a logistic regression. The predicted mortality will be used to create the surgical quality score which is the binary 30-day mortality minus the modeled risk-adjusted surgical mortality probability. | A1. Raw Inpatient file  A2. Master Beneficiary Summary File  B1. Inpatient claims  B2. Outpatient claims  B3. Ambulance claims  B4. Beneficiaries from the inpatient record with a surgical DRG code  C1. Beneficiaries from the inpatient record with a surgical DRG code  C2. File containing diagnosis information from inpatient, outpatient, and ambulance claims from the previous script  D1. Long data format of beneficiaries containing diagnosis codes from the year before the admission date  E1. Wide data format containing a comorbidity score for each beneficiary  E2. Beneficiaries from the inpatient record with a surgical DRG code  F1. Inpatient file with comorbidity, mortality, and age information  G1. Final file containing no injury codes | A1,A2. Beneficiaries from the inpatient record with a surgical DRG code  B1,B2,B3,B4. File containing diagnosis information from inpatient, outpatient, and ambulance claims for only beneficiaries in the inpatient record with a surgical DRG code  C1,C2. Long data format of beneficiaries containing diagnosis codes from the year before the admission date  D1. Wide data format containing a comorbidity score for each beneficiary  E1,E2. Inpatient file with comorbidity, mortality, and age information  F1. Final file containing no injury codes  G1. File with hospital surgical quality scores for each beneficiary |
| 19\_obtain\_hos\_qual\_scores (python) | Group by hospitals to average the surgical quality scores. Then, merge this file with the analytical files containing hospital type information (e.g. level 1, 2, …, non-trauma). This file also identifies hospitals that serve at least 90 major trauma patients within a year. After manually checking each hospital in our non-trauma sample, we remove any hospital that may have a trauma center. | 1. Final analytical file containing only hospitals with a trauma center  2. Final analytical file containing only hospitals without a trauma center  3. File with hospital surgical quality scores for each beneficiary | 1,3. Final analytical file containing only hospitals with a trauma center and including surgical quality scores  1,2. Final analytical file containing only hospitals without a trauma center and including surgical quality scores |

Notes: The numbers indicate which input files were used to create the output files. (e.g. the raw inpatient (A1) and master beneficiary summary (A2) files were used to construct the file containing beneficiaries from the inpatient record with a surgical DRG code (A1,A2))

**Step 9: Create definitions for two area types: choice and no choice**

These scripts will prepare data that calculates the distance between two hospitals or between a hospital and a census centroid using longitude and latitude coordinates. Then, we will use these datasets to create binary indicators for choice versus no choice. A hospital destination is in an area with a choice if one other hospital of a different hospital type is within a designated mile radius. On the other hand, a hospital destination is in an area with no choice if there are no surrounding hospitals within the designated mile radius. The last script in this section will obtain additional columns relating to DX codes and population parameters.

|  |  |  |  |
| --- | --- | --- | --- |
| Script name | Description | Input files | Output files |
| 20\_hos\_long\_lat  (python) | Perform a cartesian merge to link hospital with hospitals and hospital with census block centroid | 1. American Trauma Society file  2. American Hospital Association crosswalk file  3. Census block ID to long/lat coordinates (centroid) crosswalk | 1,2. Merged file with distance measure between hospitals  2,3. Merged file with distance measure between hospitals and the surrounding census block centroid |
| 21\_choice\_v\_nochoice (python) | Create indicators for two area types: choice and no choice | 1. Merged file with distance measure between hospitals  2. Final analytical file containing only hospitals with a trauma center  3. Final analytical file containing only hospitals without a trauma center | 1,2. Final analytical file containing only hospitals with a trauma center  and choice and no choice binary indicators  1,3. Final analytical file containing only hospitals without a trauma center  and choice and no choice binary indicators |
| 22\_obtain\_population\_estimate (python) | Obtain additional columns relating to DX codes and population parameters | 1. Final analytical file containing only hospitals with a trauma center  and choice and no choice binary indicators  2. Final analytical file containing only hospitals without a trauma center  and choice and no choice binary indicators  3. Various crosswalk files from AHA and county FIPS to zip.  4. Area Health Resource File (for population count on county level)  5. EPA (traffic proximity measure)  6. Merged file with distance measure between hospitals and the surrounding census block centroid | 1,3,4,5,6. Final analytical file containing only hospitals with a trauma center  and other appended columns relating to DX codes and population parameters.  2,3,4,5,6. Final analytical file containing only hospitals without a trauma center  and other appended columns relating to DX codes and population parameters. |

Notes: The numbers indicate which input files were used to create the output files.

**Step 10: Conduct regression and propensity score analysis**

The following scripts will run a logit regression and propensity score analysis and export the results to excel sheets. Note that the exported excel sheet will not contain any labels (i.e. only numbers). The labels (e.g. title, headings, footnotes, etc…) were appended separately in excel for flexibility and convenience.

|  |  |  |  |
| --- | --- | --- | --- |
| Script name | Description | Input files | Output files |
| [23\_analysis\_in\_stata](https://rcg.bsd.uchicago.edu/gitlab/sanghavi-lab/trauma_center_project/tree/trauma_center_project_jessy/python_codes/final_regression_analysis_paper_two/23_analysis_in_stata) [folder] (Stata) | This folder contains two scripts (regression\_analysis.do and pscore.do) that will analyze the final analytical files using a logit regression and propensity score overlap weighting method. Then, the scripts will generate excel tables for the table of characteristics and table with the main results | 1. Final analytical file containing only hospitals with a trauma center  and other appended columns relating to DX codes and population parameters.  2. Final analytical file containing only hospitals without a trauma center  and other appended columns relating to DX codes and population parameters. | 1,2. Table of characteristics, main results, and other datasets for figures. |

Notes: The numbers indicate which input files were used to create the output files.

**Step 11: Create figures for main exhibits**

The following script will take the dataset output from step 10 and create each figure in the main exhibits.

|  |  |  |  |
| --- | --- | --- | --- |
| Script name | Description | Input files | Output files |
| 24\_create\_graphs (python) | Creates figures 1, 2, and 3 | 1. Datasets from step 10 | 1. Figures in PDF format |

Notes: The numbers indicate which input files were used to create the output files.