

Instructions for Replicating “Tradeoffs from Integrating Diagnosis and Treatment in Markets for Health Care”

A. Data Procurement

The analyses in our paper employed databases of enrollment and insurance claim records from Medicare. Our usage of these files is governed by a data use agreement that prohibits us from sharing them with other researchers. However, the same data files are available to other researchers via an application process administered by the Centers for Medicare and Medicaid Services. The Research Data Assistance Center (ResDAC), under contract to CMS, has developed a set of application materials that should be utilized in developing an application. These materials can be found at the following web site:

http://www.resdac.umn.edu/Medicare/requesting_data_NewUse.asp

We used the following files in our analyses:

- 20% Inpatient Standard Analytic File, 1998-99: a file containing claims for all hospital stays, for a 20% sample of Medicare beneficiaries.
- 20% Outpatient Standard Analytic File, 1998-99: a file containing claims for all outpatient hospital encounters.
- 20% Medicare Provider Analysis and Review (MedPAR), 1996-2000: a file containing claims information on all inpatient stays, with claims for a given stay consolidated into a single stay-level record.
- 20% Denominator File, 1996-2000: a file containing information on the monthly enrollment histories of all Medicare beneficiaries, including whether they were enrolled in a Medicare managed care organization and their date of death if deceased.
- 20% Physician/Supplier, 1998-2000: a file containing claims for all non-institutional care (e.g., physician office visits).

In addition to these confidential files, we also used Medicare hospital cost report data (downloadable at http://www.cms.hhs.gov/CostReports/02_HospitalCostReport.asp) and files from the American Hospital Association to code characteristics of hospitals (downloadable for purchase at http://www.ahadata.com/ahadata_app/index.jsp).

B. Data analyses

We ran a series of computer programs using SAS version 8 and Stata version 9 to create our analysis datasets and run our analyses. We describe each of these programs below.

- We searched inpatient and outpatient claims from 1998 and 1999, to identify who received the three procedures analyzed in our paper: cardiac catheterization

(CATH), angioplasty and coronary artery bypass surgery (CABG). We also used these files to calculate annual hospital volumes for each of these three procedures.

- We used the MedPAR files to calculate total expenditures for hospital and skilled nursing care in the year prior and subsequent to the admission date of each patient's CATH (the "index" event).
- We combined these files with demographic (gender, age and race) and enrollment information on each patient, and removed the records of any patient who was not continuously enrolled in Medicare on a fee-for-service basis from one year before to one year after the index event. We also coded the patient's date of death if deceased.
- We coded whether the patient experienced an inpatient hospital admission for heart attack or heart failure in the two years subsequent to the index event.
- We merged data from the AHA hospital surveys and Medicare cost reports describing the following characteristics of the index event hospital. From the AHA, we coded whether the hospital was part of a system. From the cost reports, we coded the hospital's teaching status, its ownership type (non-profit, for-profit or government) and size (large versus small).
- Using the initial file of patients who received each of the three cardiac procedures, we created indicators for whether each catheterization patient underwent surgery for angioplasty and CABG.
- We searched the physician claims to create volume measures for all cardiac procedure operating physicians. We then coded each physician as a cardiac surgeon (performed one or more CABG surgeries), interventional cardiologist (performed one or more angioplasties) or non-interventional cardiologist (performed no revascularizations). We also used the physician claims to calculate expenditures on physician fees in the year preceding and following the index event.
- We used the MedPAR files to find all hospitalizations and skilled nursing stays for one year prior to the index event. We coded the diagnosis codes from these stays into a set of 96 different risk adjustment variables: 32 variables each for prior hospitalizations, prior skilled nursing stays, and index hospitalization stays.
- We used the MedPAR files to capture the first three digits of the ZIP code of treatment for each index event. We excluded cases with missing ZIP codes.
- We used the inpatient and outpatient claims to code whether the index catheterization was performed in an inpatient or outpatient setting.

Running each of these programs yielded our analysis dataset. We then performed our analyses in three steps. First, we estimate the coefficients and variance-covariance matrix for the bootstrapped multinomial logit model described in the main body of the paper. We use Stata's bootstrap command for this purpose; the procedure is run 100 times, on a bootstrapped sample the size of the full dataset. This program first runs a series of "first-stage" regressions to generate exogenous versions of our endogenous regressors (physician and hospital characteristics), using area-level averages of those variables as instruments. We then run the multinomial logit regression. Second, we perform a series of calculations on the estimates from this regression to obtain point estimates and standard errors for quantities to be used later in our decomposition. Third, we run instrumental variables models on our several outcome measures, again using our area-level measures of physician and hospital characteristics as instruments. We then perform additional calculations to generate the point estimates and standard errors for each of the quantities in the decomposition presented in table 3.

In the pages that follow, we include the Stata code for each of our three analysis programs, as well as the program that performs the statistical tests from table 2.

C. Stata program to perform bootstrapped multinomial logit regression

```
#delimit;

set mem 75m;
set more off;
set matsize 400;
display c(current_date);
display c(current_time);

use predictvars;
keep treatment
    invasivecard system teaching_hosp non_profit for_profit
    large_hosp physcathtot hospcathtot age7074 age7579 age8089
    age90plus female black acutedis1-acutedis32
    nonacutedis1-nonacutedis32 indexdis1-indexdis32
    acutehosppreexpendtot nonacutehosppreexpendtot
    acuteprehospdaystot nonacuteprehospdaystot
    area_invasivecard area_system area_teaching_hosp
    area_non_profit area_for_profit area_large_hosp
    area_physcathtot area_hospcathtot zipcode;
drop if zipcode == "";
save /tmp/predictvars, replace;
clear;
*****
****;
* DEFINE PROGRAMS TO RUN THE MNL IN TWO DIFFERENT WAYS: SINGLE AND TWO
STAGE ;
* WHERE THE CALLING OF THE REGRESSION OCCURS ;
*****
****;
program define m_n_1;
    args basic ;
    use /tmp/predictvars;
*****
**;
    * RUN THE APPROPRIATE REGRESSION ;
*****
**;
    gen treatment3 = treatment;
    replace treatment3 = 1 if treatment == 0;
    if "`basic'" == "basic" {};
        global indvar "invasivecard system teaching_hosp
non_profit for_profit large_hosp
physcathtot hospcathtot age7074 age7579
age8089 age90plus female black
acutedis1-acutedis32
nonacutedis1-nonacutedis32
indexdis1-indexdis32
acutehosppreexpendtot
nonacutehosppreexpendtot
acuteprehospdaystot
nonacuteprehospdaystot";
        mlogit treatment3 $indvar;
    };
else {};

```

```

        * RUN FIRST STAGE REGS ;
foreach dep in invasivecard system teaching_hosp non_profit
    for_profit large_hosp physcathtot hospcathtot
{
    quietly regress `dep' age7074 age7579 age8089
age90plus
        female black
        acutedis1-acutedis32
        nonacutedis1-nonacutedis32
        indexdis1-indexdis32
            acutehosppreexpendtot
            nonacutehosppreexpendtot
                acuteprehospdaystot
            nonacuteprehospdaystot
            area_invasivecard area_system
            area_teaching_hosp area_non_profit
            area_for_profit area_large_hosp
                area_physcathtot area_hospcathtot;
        predict `dep'_hat;
}
global indvar "invasivecard_hat system_hat
    teaching_hosp_hat non_profit_hat for_profit_hat
    large_hosp_hat physcathtot_hat hospcathtot_hat
    age7074 age7579 age8089 age90plus female black
    acutedis1-acutedis32
    nonacutedis1-nonacutedis32
    indexdis1-indexdis32
        acutehosppreexpendtot nonacutehosppreexpendtot
        acuteprehospdaystot nonacuteprehospdaystot";
display c(current_date);
display c(current_time);
bootstrap "mlogit treatment3 $indvar" _b, reps(100);
};

* Save vector of betas. It will be 1 x k, the number of
independent
vars plus the three cutpoint parameters;
mat betas_row=e(b);

* Turn it into a column vector;
mat betas=betas_row';

* Capture variance-covariance matrix;
mat V=e(V);

* Capture the number of categories in the dependent variable;
* Capture the number of independent variables as a scalar
variable
    and as a local variable;
if "`basic'" == "basic" {};
    scalar cats=e(k_cat);
    scalar df_m=e(df_m);
};
else {
    * I HAVE HARD CODED THIS TO WORK WITH THE BOOTSTRAP;
    scalar cats=3;
    scalar df_m=(rowsof(betas))-2;
}

```

```

};

* Capture the number of independent variables as a scalar
variable
    and as a local variable;
local cats=cats;
scalar k=(df_m)/(cats-1)+1;
local numvar=(df_m)/(cats-1);

* Mark patients in this estimation sample;
mark include if e(sample)==1;

* Create a vector of means, to be used for calculating predicted
values and SEs of predictions.
First, create an empty column vector for means;
mat means=J(k,1,1);

* Populate the means vector;
for VAR in var $indvar \ NUM in num 1/`numvar', noheader :
    quietly summ VAR if include==1 \
    mat means[NUM,1]=r(mean);

* Drop the dataset so that you don't have to worry about memory
(the vectors and matrices defined so far will be retained);
drop _all;

* Convert the vector of means into a dataset;
cd /dir64/ankur/tmp/decomp;
svmat double means, name(means);
save means_`basic', replace;
drop _all;

* Save the betas to a dataset;
svmat double betas, name(betas);
save betas_`basic', replace;
drop _all;

* Save the variance-covariance matrix to a dataset;
svmat double V, name(V);
save V_`basic', replace;
drop _all;

end;

*****
****;
* RUN THE PROGRAMS WE HAVE DEFINED ABOVE ;
*****
****;
m_n_l basic;
m_n_l boot;

display c(current_date);
display c(current_time);
clear;
exit;

```

D. Stata program to calculate inputs for decomposition, using outputs from multinomial logit

```
#delimit;

set mem 75m;
set more off;
set matsize 400;
display c(current_date);
display c(current_time);

program define patientshare;
    args basic ;
    if "`basic'" == "basic" {
        global indvar "invasivecard system teaching_hosp
                      non_profit for_profit large_hosp
                      physcathtot hospcathtot age7074 age7579
                      age8089 age90plus female black
                      acutedis1-acutedis32
                      nonacutedis1-nonacutedis32
                      indexdis1-indexdis32
                      acutehosppreexpendtot
                      nonacutehosppreexpendtot
                      acuteprehospdaystot
                      nonacuteprehospdaystot";
    }
    else {
        global indvar "invasivecard_hat system_hat
                      teaching_hosp_hat non_profit_hat for_profit_hat
                      large_hosp_hat physcathtot_hat hospcathtot_hat
                      age7074 age7579 age8089 age90plus female black
                      acutedis1-acutedis32
                      nonacutedis1-nonacutedis32
                      indexdis1-indexdis32
                      acutehosppreexpendtot nonacutehosppreexpendtot
                      acuteprehospdaystot nonacuteprehospdaystot";
    }
}

cd /tmp;

* Read in means dataset, turn it into a vector;
use means_`basic'.dta;
mkmat means1, matrix(means);

* Next, create a matrix of x values. This will have the same
number of rows as independent variables (plus one at
the end for the constant), and two columns (because
we will generate a separate set of patient share
predictions for interventionist and non-interventionist
patients);
scalar predictions=2;
local predictions=predictions;
mat xmatrix=J(rowsof(means),predictions,1);

* Set each column equal to the vector of means;
```

```

for NUM in num 1/`predictions',noheader: mat
xmatrix[1,NUM]=means;

* Make changes to individual elements of each vector: the first
prediction is for invasive cardiologists,
and the second is for noninvasive docs;
mat xmatrix[1,1]=1;
mat xmatrix[1,2]=0;

* Name rows and columns of the matrix -- this will make it easier
to change individual elements of the
matrix later on;
matrix rownames xmatrix=$indvar constant;
matrix colnames xmatrix=inv noninv;

* Convert xmatrix into a dataset and then into a comma delimited
file, so that you can easily open it
and make sure everything is correct;
svmat double xmatrix;
cd /tmp/decomp;
outfile xmatrix1-xmatrix`predictions'
using xmatrix_mnl_`basic'.csv, comma wide replace;

* Drop all data so the next dataset can be read in;
drop _all;

* Read in the beta dataset, turn it into a vector;
use betas_`basic'.dta;
mkmat betas1, matrix(betas);

* Use the number of rows in the means vector and the beta vector
to define the number of independent
variables and number of DV categories;
scalar k=rowsof(means);
scalar cats=rowsof(betas)/k+1;

* Create a predictions matrix to hold to the predictions and
their standard errors;
mat preds=J(predictions*cats,2,0);

* Break up the vector of betas saved earlier into three separate
vectors, one for each category;
local i=1;
while `i'<=cats-1 {
    mat betas`i'=betas[(`i'-1)*k+1..`i'*k,1];
    local i=`i'+1;
};

* Calculate the predicted patient shares;
local i=1;
while `i'<=predictions {
    local j=1;
    local denom=1;
    * First calculate the x*beta for each pairwise comparison,
    and then transform them into logit probabilities;
    while `j'<=cats-1 {
        mat xb`j'=xmatrix[1...,`i']'*betas`j';
    };
    local j=2;
    while `j'<=cats {
        local denom=denom+xb`j';
        mat xb`j'=xb`j'/denom;
        local j=+1;
    };
};

```

```

        scalar xb`j'=el(xb`j',1,1);
        local denom=`denom'+exp(xb`j');
        local j=`j'+1;
    };
local j=1;
* Each prediction is equal to exp(xb[i])/  

* (1+exp(xb[1]+...+exp(xb[cats-1])),  

* where i subscripts choice being compared with the  

* reference choice;
while `j'<=cats-1 {
    mat preds[`j'+cats*(`i'-1),1]=exp(xb`j')/`denom';
    local j=`j'+1;
};
* The exception is the reference choice, which is the  

* category with the highest number:  

* 1/(1+exp(xb[1]+...+exp(xb[cats-1])));
mat preds[cats*`i',1]=1/`denom';
local i=`i'+1;
};

* Check the preds matrix to see what it looks like;
* DO THIS AT THE END ;
*mat list preds;

* Drop all data so the next dataset can be read in;
drop _all;

* Read in the variance-covariance matrix dataset, turn it into a
matrix;
local 3numvar=k*(cats-1);
use V_basic.dta;
mkmata V1-V`3numvar', matrix(V);

* Next, calculate gradient vectors. Each gradient vector will
be (cats-1)*k by 1 -- a k by 1 vector for the independent
variables from each pairwise comparison. Each subvector
is the gradient with respect to the betas associated with
one of the pairwise comparisons from the MNL, so there will
be 3 gradients to calculate for each of the eight
predictions;

* Start by calculating the gradients (the pdfs) of the logit cdf.
Store them in a matrix that's has the same number of rows
as categories, and the same number of columns as
predictions;
mat pdf=J(cats-1,cats*predictions,0);
local i=1;
while `i'<=predictions {
    local j=1;
    local denom=1;
    while `j'<=cats-1 {
        mat xb`j'=xmatrix[1...,`i']'*betas`j';
        scalar xb`j'=el(xb`j',1,1);
        local denom=`denom'+exp(xb`j');
        local j=`j'+1;
    };
    * Set the gradients for predictions 1-3 and 5-7;
}

```

```

local j=1;
while `j'<=cats-1 {
    * First, set all three rows of the column you're
        working on to be the "default" gradient
        WE ARE NOW USING ONLY 3 CATEGORIES SO WE
        WILL ONLY HAVE XB1 AND XB2 ;
    mat pdf[1,`j'+cats*(`i'-1)]=-exp(xb`j'+xb1)/
        (`denom')^2;
    mat pdf[2,`j'+cats*(`i'-1)]=-exp(xb`j'+xb2)/
        (`denom')^2;
    *mat pdf[3,`j'+cats*(`i'-1)]=-exp(xb`j'+xb3)/
        (`denom')^2;
    * Then set one of the three rows to the
        "non-default" gradient -- row 1 in
        columns 1 and 5, row 2 in 2 and 6,
        row 3 in 3 and 7;
    mat pdf[`j',`j'+cats*(`i'-1)]=(exp(xb`j'))*(`denom')
        -exp(2*xb`j'))/(`denom')^2;
    local j=`j'+1;
};
* Set the gradients for predictions 4 and 8;
local j=1;
while `j'<=cats-1 {
    mat pdf[`j',cats*`i']= -exp(xb`j')/(`denom')^2;
    local j=`j'+1;
};
local i=`i'+1;
};

* Next, create the gradient vectors;
* Initialize a matrix that will hold the vectors for each
    prediction (# of rows is k times the # of cats
    minus 1, # of cols is # of cats times number of
    predictions);
mat grad_vecs_old=J((cats-1)*k,cats*predictions,0);

* Take the kronecker product of the gradient matrix from above
and
the xmatrix -- this will create our
matrix of gradients vectors;
local i=1;
while `i'<=predictions {
    mat grad_vecs_old[1,1+cats*(`i'-1)]=
        pdf[1...,1+cats*(`i'-1)..cats+cats*(`i'-1)]
        #xmatrix[1...,`i'];
    local i=`i'+1;
};

* Since we will later be looking at the weighted sum of the
positive
of the first four columns and the negative of the last four
columns of this gradient vectors matrix, modify the matrix
so that the last four columns are set to their negative;
* AGAIN, THERE ARE NO LONGER 8 COLUMNS/TREATMENT PATHS, BUT
6. ;
scalar neg_one=-1;
mat grad_vecs=grad_vecs_old;

```

```

mat grad_vecs[1,4]=grad_vecs_old[1...,4..6]*neg_one;

* Create variance-covariance matrix for predicted patient shares
   in the preds vector;
mat variance=grad_vecs'*V*grad_vecs;

* Get the main diagonal from this matrix. This will be the
vector
   of variances, one for each patient share prediction. Add
   to the preds matrix;
mat preds[1,2]=vecdiag(variance)';

* Construct three additional vectors. Two are the differences
   between pairs of vectors;
* a) P[cabg|noninvasive] - P[cabg|invasive] : column 7 minus
      column 3;
* b) P[mm|noninvasive] - P[mm|invasive] : column 8 minus column
4;
* CHANGE HERE FOR 8 TO 6 TREATMENTS ;
mat grad_diff_vecs=J(rowsof(grad_vecs),3,0);
mat grad_diff_vecs[1,1]=grad_vecs_old[1...,5]-
grad_vecs_old[1...,2];
mat grad_diff_vecs[1,2]=grad_vecs_old[1...,6]-
grad_vecs_old[1...,3];
* The third vector is the weighted average of two vectors;
* c) P[ptca self_refer|invasive, PTCA] : column 1 divided by the
   sum of column 1 and column 2;
* DOING C THIS WAY NO LONGER MAKES SENSE SINCE THERE IS NO
   LONGER A DISTINCTION BETWEEN SELF AND OUT REFERRAL.
   SO WE SHOULD BE ABLE TO TREAT THIS FINAL DIFFERENCE
   THE SAME WE MIGHT THE TWO DIFFERENCES ABOVE .;
*mat pred_ptca_self=preds[1,1]/(preds[1,1]+preds[2,1]);
*mat grad_diff_vecs[1,3]=pred_ptca_self*grad_vecs_old[1...,1]+
   (1-pred_ptca_self)*(grad_vecs_old[1...,2]);
* I THINK FOR THE NEW VERSION OF THE DECOMP, WE WONT
   EVEN NEED THIS DIFFERENCE ;
mat grad_diff_vecs[1,3]=grad_vecs_old[1...,4]-
grad_vecs_old[1...,1];

* Create predicted values and variances for differences;
mat preds_diff=J(cols of(grad_diff_vecs),2,0);
mat preds_diff[1,1]=preds[5,1]-preds[2,1];
mat preds_diff[2,1]=preds[6,1]-preds[3,1];
* AGAIN THIS IS SIMPLIFIED ;
*mat preds_diff[3,1]=preds[1,1]/(preds[1,1]+preds[2,1]);
mat preds_diff[3,1]=preds[4,1]-preds[1,1];

mat variance_diff=grad_diff_vecs'*V*grad_diff_vecs;
mat preds_diff[1,2]=vecdiag(variance_diff)';

mat list preds;
mat list preds_diff;

cd /tmp;

* Save the two predictions-se matrices to .csv files;
svmat double preds;

```

```

outfile preds1-preds2
    using preds_mnl_`basic'.csv, comma wide replace;

svmat double preds_diff;
outfile preds_diff1-preds_diff2
    using preds_diff_mnl_`basic'.csv, comma wide replace;

* Save the variance-covariance matrix for the predictions to a
    .csv file;
svmat double variance;
outfile variance1-variance6
    using V_preds_mnl_`basic'.csv, comma wide replace;

clear;

end;

*****
****;
* RUN PROGRAM
;
*****
****;
patientshare basic;
patientshare boot;

display c(current_date);
display c(current_time);
clear;
exit;

```

E. Stata program to run output regressions and calculate final decomposition outputs

```
#delimit;

set mem 75m;
set more off;
set matsize 400;
display c(current_date);
display c(current_time);

use predictvars;

keep hospclaims postexpend physclaims postexpend postexpend ami365day
    hf365day dead365day treatment invasivecard system teaching_hosp
    non_profit for_profit large_hosp physcathtot hospcathtot age7074
    age7579 age8089 age80plus female black acutedis1-acutedis32
    nonacutedis1-nonacutedis32 indexdis1-indexdis32
    acutehosppreexpendtot nonacutehosppreexpendtot
    acuteprehospdaystot nonacuteprehospdaystot
    area_invasivecard area_system area_teaching_hosp
    area_non_profit area_for_profit area_large_hosp
    area_physcathtot area_hospcathtot
    cabgphyscabgvolume cabghospcabgvolume
    ptcaphysptcavolume ptcahospptcavolume zipcode;
drop if zipcode == "";

*****
****;
* RECALCULATE AREA VARIABLES FOR CABG AND PTCA VOLUME;
*****
****;

sort zipcode;
save ivreg_temp_master, replace;
* MAKE AREA AVERAGES FOR ONLY THOSE PROVIDERS THAT RECIEVED A PTCA
    PATIENT;
keep if treatment == 0 | treatment == 1;
collapse (mean) ptcaphysptcavolume ptcahospptcavolume, by (zipcode);
rename ptcaphysptcavolume area_ptcaphysptcavolume;
rename ptcahospptcavolume area_ptcahospptcavolume;
keep zipcode area_ptcaphysptcavolume area_ptcahospptcavolume;
sort zipcode;
save ivreg_temp_ptca, replace;
clear;
* MAKE CABG AREA VOLUME FOR ONLY THOSE PROVIDERS THAT RECIEVED A CABG
    PATIENT;
use ivreg_temp_master;
keep if treatment == 2;
collapse (mean) cabgphyscabgvolume cabghospcabgvolume, by (zipcode);
rename cabgphyscabgvolume area_cabgphyscabgvolume;
rename cabghospcabgvolume area_cabghospcabgvolume;
keep zipcode area_cabgphyscabgvolume area_cabghospcabgvolume;
sort zipcode;
save ivreg_temp_cabg, replace;
clear;
* MERGE EVERYTHING TOGETHER ;
```

```

use ivreg_temp_master;
merge zipcode using ivreg_temp_ptca;
drop _merge;
sort zipcode;
merge zipcode using ivreg_temp_cabg;
drop _merge;
* THOSE ZIP CODES WITH ZERO PTCAS OR CABGS GET VOLUMES OF ZERO ;
foreach var of varlist area_ptcaphysptcavolume area_ptcahosptcavolume
    area_cabgphyscabgvolume area_cabghospcabgvolume {};
    replace `var' = 0 if `var' == .;
};
save predictvars_decomp, replace;
clear;

*****
* NOW WE CAN START WITH THE DECOMPOSITION ;
*****
* Initialize program to create predictions and standard errors;
program define decomp;
    args depvar int basic;

    * Read in matrices of predicted patient shares, variances,
        predicted differences -- make sure you read in the right
        ones for this version. Make sure that you read in the
        correct number of rows of each matrix -- sometimes there
        are extra empty rows ;

    * Predicted patient shares and their variances ;
    infile predshare1 predshare2 using
        "preds_mnl_`basic'.csv" in 1/6;
    mkmat predshare1-predshare2;
    mat predshare=J(6,2,0);
    mat predshare[1,1]=predshare1[1...,1];
    mat predshare[1,2]=predshare2[1...,1];
    matrix rownames predshare = inv_ptca inv_cabg inv_medmg
        noninv_ptca noninv_cabg noninv_medmgmt;
    mat list predshare;
    drop _all;

    * Predicted differences and their variances ;
    infile predshdiff1 predshdiff2 using
        "preds_diff_mnl_`basic'.csv"
        in 1/3;
    mkmat predshdiff1-predshdiff2;
    mat predshare_diff=J(3,2,0);
    mat predshare_diff[1,1]=predshdiff1[1...,1];
    mat predshare_diff[1,2]=predshdiff2[1...,1];
    matrix rownames predshare_diff = cabg medmgmt ptca;
    mat list predshare_diff;
    drop _all;

    * Variance-covariance matrix for predictions ;
    local i=6;
    infile var1-var`i' using
        "V_preds_mnl_`basic'.csv" in 1/6;

```

```

mkmat var1-var`i';

mat Vshare=var1;

local j=2;
while `j'<=`i' {
    mat Vshare=Vshare,var`j'[1...,1];
    local j=`j'+1;
};

mat list Vshare;

drop _all;

use predictvars_decomp;
cd decomp;

*****
***;
* CREATE THE INTERACTION VARIABLES (AND GROUP THEM INTO HANDY
MACROS) WE NEED IN THE IV REG ;
*****;
***;

gen ptca = 0;
quietly replace ptca = 1 if treatment == 1 | treatment == 0;
gen cabg = 0;
quietly replace cabg = 1 if treatment == 2;
global physhosp "invasivecard system teaching_hosp non_profit
for_profit large_hosp physcathtot
hospcathtot ptcaphysptcavolume
ptcahospptcavolume cabgphyscabgvolume
cabghospcabgvolume";
foreach num of numlist 1 2 {};
foreach ins of global physhosp {};
gen `ins'_`num' = 0;
quietly replace `ins'_`num' = `ins' if treatment
== `num';
};
global area "area_invasivecard area_system area_teaching_hosp
area_non_profit area_for_profit
area_large_hosp area_physcathtot
area_hospcathtot
area_ptcaphysptcavolume
area_ptcahospptcavolume
area_cabgphyscabgvolume
area_cabghospcabgvolume";
foreach num of numlist 1 2 {};
foreach ins of global area {};
gen `ins'_`num' = 0;
quietly replace `ins'_`num' = `ins' if treatment
== `num';
};
rename invasivecard_1 inv_ptca;
rename invasivecard_2 inv_cabg;
global individual "ptca cabg"

```

```

        age7074 age7579 age8089 age90plus female
        black acutedisl-acutedis32
        nonacutedisl-nonacutedis32
        indexdisl-indexdis32
        acutehosppreexpendtot
        nonacutehosppreexpendtot
        acuteprehospdaystot
        nonacuteprehospdaystot" ;
global interact "inv_ptca system_1 teaching_hosp_1 non_profit_1
                for_profit_1 large_hosp_1 physcathtot_1
                hospcathtot_1
                inv_cabg system_2 teaching_hosp_2 non_profit_2
                for_profit_2 large_hosp_2 physcathtot_2
                hospcathtot_2 ";
global interact_a "area_invasivecard_1 area_system_1
                  area_teaching_hosp_1 area_non_profit_1
                  area_for_profit_1 area_large_hosp_1
                  area_physcathtot_1 area_hospcathtot_1
                  area_invasivecard_2 area_system_2
                  area_teaching_hosp_2 area_non_profit_2
                  area_for_profit_2 area_large_hosp_2
                  area_physcathtot_2 area_hospcathtot_2";

*****
***;
* IF THE ARGUMENT INT IS 1 WE WANT TO USE INTERACTIONS. IF IT IS
ZERO WE DO NOT.
HERE BE SOME FANCY PROGRAMMING TO AUTOMATE THAT - ANYHOW -
HERE WE ARE RUNING THE IVREGS ;
* NOTE THAT WE ARE NOT PASSING PREDICTED VALUES FOR THE
INDEPENDENT
VARS TO THE REST OF THE PROGRAM.
WE JUST WANT THE MEANS OF THESE VARS, THE PREDICTED MEANS
SHOULD BE THE SAME AS THE MEANS OF THE ORIGINAL VALUES ;
*****
***;
display c(current_date);
display c(current_time);
if "`int'" == "int" {
    if "`basic'" == "basic" {
        reg `depvar' $physhosp $interact $individual;
    };
    else {
        ivreg `depvar' ($physhosp $interact =
                      $area $interact_a) $individual, robust;
    };
    * Define string of independent variables ;
    global indvar "$physhosp $interact $individual";
    };
}
else {
    if "`basic'" == "basic" {
        reg `depvar' $physhosp $individual;
    };
    else {
        ivreg `depvar' ($physhosp = $area) $individual,
        robust;
    };
}

```

```

        * Define string of independent variables ;
        global indvar "$physhosp $individual";
};

* Save vector of betas. It will be 1 x k, the number of
    independent variables (including the constant) ;
mat betas_row=e(b);

* Turn it into a column vector ;
mat betas=betas_row';

* Save the variance-covariance matrix. It will be k x k. ;
mat V=e(V);

* Capture the number of independent variables as a scalar
variable
    and as a local variable. For the scalar, add one for the
    constant. ;
scalar k=e(df_m)+1;
local numvar=e(df_m);

* mark patients in this estimation sample ;
mark include if e(sample)==1;

* Create matrix of means for predicted values. First, create
empty
    column vector for means ;
mat means=J(k,1,1);

* Populate the means vector. Leave the last row alone, remains at
    1 for the constant ;
for VAR in var $indvar \ NUM in num 1/\`numvar',noheader:
    quietly summ VAR if include==1 \
    mat means[NUM,1]=r(mean);

* drop the dataset so that you dont have to worry about memory
    (the vectors and matrices defined so far will be retained)
;
drop _all;

* Create a matrix of x values. should have the same number of
rows
    as independent variables, and the same number of columns
    as predictions you want to make ;
scalar predictions=8;
local predictions=predictions;
mat xmatrix=J(k,predictions,1);

* Set each column equal to the vector of means ;
for NUM in num 1/\`predictions',noheader: mat
xmatrix[1,NUM]=means;

* Name rows this will make it easier to change individual
elements
    of the matrix later on ;
matrix rownames xmatrix=$indvar;

```

```

* Name columns too ;
matrix colnames xmatrix=
    all_inv inv_ptca inv_cabg inv_medmgmt
    all_noninv noninv_ptca noninv_cabg noninv_medmgmt;

* Make changes to individual elements of each vector ;
mat xmatrix[rownumb(xmatrix,"invasivecard"),
    colnumb(xmatrix,"all_inv")]=J(1,4,1);
mat xmatrix[rownumb(xmatrix,"invasivecard"),
    colnumb(xmatrix,"all_noninv")]=J(1,4,0);

mat xmatrix[rownumb(xmatrix,"ptca"),
    colnumb(xmatrix,"inv_ptca")]=1;
mat xmatrix[rownumb(xmatrix,"ptca"),
    colnumb(xmatrix,"noninv_ptca")]=1;
mat xmatrix[rownumb(xmatrix,"ptca"),
    colnumb(xmatrix,"inv_cabg")]=J(1,2,0);
mat xmatrix[rownumb(xmatrix,"ptca"),
    colnumb(xmatrix,"noninv_cabg")]=J(1,2,0);

mat xmatrix[rownumb(xmatrix,"cabg"),
    colnumb(xmatrix,"inv_ptca")]=J(1,3,0);
mat xmatrix[rownumb(xmatrix,"cabg"),
    colnumb(xmatrix,"inv_cabg")]=1;
mat xmatrix[rownumb(xmatrix,"cabg"),
    colnumb(xmatrix,"noninv_ptca")]=J(1,3,0);
mat xmatrix[rownumb(xmatrix,"cabg"),
    colnumb(xmatrix,"noninv_cabg")]=1;

* I think we need to adjust the means for the cabg/ptca volume
variables. The reason is that these are actually
interaction variables -- zero when the treatment is absent, and the
volume mean if the treatment is present. So we should be
setting the mean to zero for cases where the ptca/cabg
treatment was not applied, and the mean of the
variable for cases when the treatment was applied ;
mat xmatrix[rownumb(xmatrix,"ptcaphysptcavolume"),
    colnumb(xmatrix,"inv_cabg")]=J(1,2,0);
mat xmatrix[rownumb(xmatrix,"ptcaphysptcavolume"),
    colnumb(xmatrix,"noninv_cabg")]=J(1,2,0);
mat xmatrix[rownumb(xmatrix,"ptcahosptcavolume"),
    colnumb(xmatrix,"inv_cabg")]=J(1,2,0);
mat xmatrix[rownumb(xmatrix,"ptcahosptcavolume"),
    colnumb(xmatrix,"noninv_cabg")]=J(1,2,0);
mat xmatrix[rownumb(xmatrix,"cabgphyscabgvolume"),
    colnumb(xmatrix,"inv_ptca")]=J(1,1,0);
mat xmatrix[rownumb(xmatrix,"cabgphyscabgvolume"),
    colnumb(xmatrix,"inv_medmgmt")]=J(1,1,0);
mat xmatrix[rownumb(xmatrix,"cabgphyscabgvolume"),
    colnumb(xmatrix,"noninv_ptca")]=J(1,1,0);
mat xmatrix[rownumb(xmatrix,"cabgphyscabgvolume"),
    colnumb(xmatrix,"noninv_medmgmt")]=J(1,1,0);
mat xmatrix[rownumb(xmatrix,"cabghospcabgvolume"),
    colnumb(xmatrix,"inv_ptca")]=J(1,1,0);
mat xmatrix[rownumb(xmatrix,"cabghospcabgvolume"),
    colnumb(xmatrix,"inv_medmgmt")]=J(1,1,0);

```

```

mat xmatrix[rownumb(xmatrix,"cabghospcabgvolume"),
            colnumb(xmatrix,"noninv_ptca")]=J(1,1,0);
mat xmatrix[rownumb(xmatrix,"cabghospcabgvolume"),
            colnumb(xmatrix,"noninv_medmgmt")]=J(1,1,0);

* THE CODE BELOW IS RESPONSIBLE FOR VALUES WITH THE INTERACTION
REGRESSION. I HOPE ITS NOT CONFUSING. ;
if "`int'" == "int" {
    mat xmatrix[rownumb(xmatrix,"inv_ptca"),
                colnumb(xmatrix,"all_inv")]=
        means[rownumb(xmatrix,"ptca"),1];
    mat xmatrix[rownumb(xmatrix,"inv_ptca"),
                colnumb(xmatrix,"inv_ptca")]=J(1,7,0);
    mat xmatrix[rownumb(xmatrix,"inv_ptca"),
                colnumb(xmatrix,"inv_ptca")]=1;

    mat xmatrix[rownumb(xmatrix,"inv_cabg"),
                colnumb(xmatrix,"all_inv")]=
        means[rownumb(xmatrix,"cabg"),1];
    mat xmatrix[rownumb(xmatrix,"inv_cabg"),
                colnumb(xmatrix,"inv_ptca")]=J(1,7,0);
    mat xmatrix[rownumb(xmatrix,"inv_cabg"),
                colnumb(xmatrix,"inv_cabg")]=1;

foreach var in system teaching_hosp non_profit
    for_profit large_hosp physcathtot hospcathtot
};

    mat xmatrix[rownumb(xmatrix,"`var'_1"),
                colnumb(xmatrix,"inv_ptca")]=J(1,3,0);
    mat xmatrix[rownumb(xmatrix,"`var'_1"),
                colnumb(xmatrix,"noninv_ptca")]=J(1,3,0);
    mat xmatrix[rownumb(xmatrix,"`var'_1"),
                colnumb(xmatrix,"inv_ptca")]=
        means[rownumb(xmatrix,"`var'"),1];
    mat xmatrix[rownumb(xmatrix,"`var'_1"),
                colnumb(xmatrix,"noninv_ptca")]=
        means[rownumb(xmatrix,"`var'"),1];
};

foreach var in system teaching_hosp non_profit
    for_profit large_hosp physcathtot hospcathtot
};

    mat xmatrix[rownumb(xmatrix,"`var'_2"),
                colnumb(xmatrix,"inv_ptca")]=J(1,3,0);
    mat xmatrix[rownumb(xmatrix,"`var'_2"),
                colnumb(xmatrix,"noninv_ptca")]=J(1,3,0);
    mat xmatrix[rownumb(xmatrix,"`var'_2"),
                colnumb(xmatrix,"inv_cabg")]=
        means[rownumb(xmatrix,"`var'"),1];
    mat xmatrix[rownumb(xmatrix,"`var'_2"),
                colnumb(xmatrix,"noninv_cabg")]=
        means[rownumb(xmatrix,"`var'"),1];
};

};

* convert xmatrix into a dataset and then into a comma delimited
file, to make sure everything is correct ;

```

```

cd /dir64/ankur/tmp/decomp;
svmat double xmatrix;
outfile xmatrix1-xmatrix`predictions'
    using xmatrix_outcome_`depvar'`basic'_`int'.csv,
    comma wide replace;

* Create a matrix that will hold both the predictions and
their variances.
*Subtract one from
    the number of rows, because we're going to want nine rows
in this matrix -- like the mnl preds matrix, plus a row
for inv PTCA. WEVE CHANGED THIS. THIS SHOULD HAVE THE
SAME NUMBER OF COLUMNS AS THE MNL PREDs SINCE WE COMBINED
THE PTCA GROUPS ;
mat preds=J(predictions,2,0);

* Name rows ;
matrix rownames preds=
    all_inv inv_ptca inv_cabg inv_medmgmt
    all_noninv noninv_ptca noninv_cabg noninv_medmgmt;

* First, calculate predictions;
mat preds[1,1]=xmatrix'*betas;

* Calculate variances;
mat variance=xmatrix'*V*xmatrix;

* Get the main diagonal from this matrix. This will be the
vector
    of variances, one for each prediction. Add to the preds
matrix;
mat preds[1,2]=vecdiag(variance)';

* Print this out to make sure it looks ok;
mat list preds;
cd /dir64/ankur/tmp/decomp;
svmat double preds;
outfile predsl-preds2
    using preds_outcome_`depvar'`basic'_`int'.csv,
    comma wide replace;

* Create predicted values and variances for several differences
-- there will be seven of them total
    CHANGED THIS TO 5, DUE TO THE CHANGE IN PTCA;
mat preds_diff=J(6,2,0);
mat diff_vecs=J(k,6,0);
matrix rownames preds_diff = inv_noninv inv_ptca_cabg
    inv_ptca_medmgmt ptca cabg medmgmt;
matrix colnames diff_vecs = inv_noninv inv_ptca_cabg
    inv_ptca_medmgmt ptca cabg medmgmt;

* The first group of two represents differences between PTCA
and the two other treatment paths, for invasive patients
only ADDED A TOP ROW FOR OVERALL ;
mat preds_diff[rownumb(preds_diff,"inv_noninv"),1]=
    preds[rownumb(preds,"all_inv"),1]-
    preds[rownumb(preds,"all_noninv"),1];

```

```

mat preds_diff[rownumb(preds_diff,"inv_ptca_cabg"),1]=
    preds[rownumb(preds,"inv_ptca"),1]-
    preds[rownumb(preds,"inv_cabg"),1];
mat preds_diff[rownumb(preds_diff,"inv_ptca_medmgmt"),1]=
    preds[rownumb(preds,"inv_ptca"),1]-
    preds[rownumb(preds,"inv_medmgmt"),1];

mat diff_vecs[1,colnumb(diff_vecs,"inv_noninv")]=
    xmatrix[1...,colnumb(xmatrix,"all_inv")]-
    xmatrix[1...,colnumb(xmatrix,"all_noninv")];
mat diff_vecs[1,colnumb(diff_vecs,"inv_ptca_cabg")]=
    xmatrix[1...,colnumb(xmatrix,"inv_ptca")]-
    xmatrix[1...,colnumb(xmatrix,"inv_cabg")];
mat diff_vecs[1,colnumb(diff_vecs,"inv_ptca_medmgmt")]=
    xmatrix[1...,colnumb(xmatrix,"inv_ptca")]-
    xmatrix[1...,colnumb(xmatrix,"inv_medmgmt")];

* The second group of three represents differences
  between invasive and noninvasive patients;
mat preds_diff[rownumb(preds_diff,"ptca"),1]=
    preds[rownumb(preds,"inv_ptca"),1]-
    preds[rownumb(preds,"noninv_ptca"),1];
mat preds_diff[rownumb(preds_diff,"cabg"),1]=
    preds[rownumb(preds,"inv_cabg"),1]-
    preds[rownumb(preds,"noninv_cabg"),1];
mat preds_diff[rownumb(preds_diff,"medmgmt"),1]=
    preds[rownumb(preds,"inv_medmgmt"),1]-
    preds[rownumb(preds,"noninv_medmgmt"),1];

mat diff_vecs[1,colnumb(diff_vecs,"ptca")]=
    xmatrix[1...,colnumb(xmatrix,"inv_ptca")]-
    xmatrix[1...,colnumb(xmatrix,"noninv_ptca")];
mat diff_vecs[1,colnumb(diff_vecs,"cabg")]=
    xmatrix[1...,colnumb(xmatrix,"inv_cabg")]-
    xmatrix[1...,colnumb(xmatrix,"noninv_cabg")];
mat diff_vecs[1,colnumb(diff_vecs,"medmgmt")]=
    xmatrix[1...,colnumb(xmatrix,"inv_medmgmt")]-
    xmatrix[1...,colnumb(xmatrix,"noninv_medmgmt")];

* ASSIGN THE DIFFERENCES VARIANCES ;
mat variance_diff=diff_vecs'*V*diff_vecs;
mat preds_diff[1,2]=vecdiag(variance_diff)';

* Print this out to make sure it looks ok;
mat list preds_diff;

* Create matrix to hold elements of decomposition and their
  standard errors;
* It will have eight elements: one for the total difference,
  five for the components of the invasive-noninvasive
  decomp, and two for the elements of the PTCA invasive-
  noninvasive decomp THIS HAS BEEN REDUCED TO 6.
  CHECK OUT THE REFERRALS PAPER FOR DETAILS.;

mat decomp=J(6,2,0);
mat rownames decomp = overall cabgtoptca medmgmttoptca exp_ptca
    exp_cabg exp_medmgmt;

```

```

* a) Diversion of CABG patients to PTCA;
mat decomp[rownumb(decomp,"cabgtoptca"),1]=
    predshare_diff[rownumb(predshare_diff,"cabg"),1]*
    preds_diff[rownumb(preds_diff,"inv_ptca_cabg"),1];
mat
vartemp=(preds_diff[rownumb(preds_diff,"inv_ptca_cabg"),1]^2)*
    predshare_diff[rownumb(predshare_diff,"cabg"),2]+
    (predshare_diff[rownumb(predshare_diff,"cabg"),1]^2)*
    preds_diff[rownumb(preds_diff,"inv_ptca_cabg"),2];
mat decomp[rownumb(decomp,"cabgtoptca"),2]=sqrt(el(vartemp,1,1));

* b) Diversion of medical management patients to PTCA;
mat decomp[rownumb(decomp,"medmgmttoptca"),1]=
    predshare_diff[rownumb(predshare_diff,"medmgmt"),1]*
    preds_diff[rownumb(preds_diff,"inv_ptca_medmgmt"),1];
mat
vartemp=(preds_diff[rownumb(preds_diff,"inv_ptca_medmgmt"),1]^2)*
    predshare_diff[rownumb(predshare_diff,"medmgmt"),2]+
    (predshare_diff[rownumb(predshare_diff,"medmgmt"),1]^2)*
    preds_diff[rownumb(preds_diff,"inv_ptca_medmgmt"),2];
mat
decomp[rownumb(decomp,"medmgmttoptca"),2]=sqrt(el(vartemp,1,1));

* c) Difference in invasive/non-invasive expenditure for PTCA;
mat decomp[rownumb(decomp,"exp_ptca"),1]=
    predshare[rownumb(predshare,"noninv_ptca"),1]*
    preds_diff[rownumb(preds_diff,"ptca"),1];
mat vartemp=(preds_diff[rownumb(preds_diff,"ptca"),1]^2)*
    predshare[rownumb(predshare,"noninv_ptca"),2]+
    (predshare[rownumb(predshare,"noninv_ptca"),1]^2)*
    preds_diff[rownumb(preds_diff,"ptca"),2];
mat decomp[rownumb(decomp,"exp_ptca"),2]=sqrt(el(vartemp,1,1));

* d) Difference in invasive/non-invasive expenditure for CABG;
mat decomp[rownumb(decomp,"exp_cabg"),1]=
    predshare[rownumb(predshare,"noninv_cabg"),1]*
    preds_diff[rownumb(preds_diff,"cabg"),1];
mat vartemp=(preds_diff[rownumb(preds_diff,"cabg"),1]^2)*
    predshare[rownumb(predshare,"noninv_cabg"),2]+
    (predshare[rownumb(predshare,"noninv_cabg"),1]^2)*
    preds_diff[rownumb(preds_diff,"cabg"),2];
mat decomp[rownumb(decomp,"exp_cabg"),2]=sqrt(el(vartemp,1,1));

* e) Difference in invasive/non-invasive expenditure for medical
management;
mat decomp[rownumb(decomp,"exp_medmgmt"),1]=
    predshare[rownumb(predshare,"noninv_medmgmt"),1]*
    preds_diff[rownumb(preds_diff,"medmgmt"),1];
mat vartemp=(preds_diff[rownumb(preds_diff,"medmgmt"),1]^2)*
    predshare[rownumb(predshare,"noninv_medmgmt"),2]+
    (predshare[rownumb(predshare,"noninv_medmgmt"),1]^2)*
    preds_diff[rownumb(preds_diff,"medmgmt"),2];
mat
decomp[rownumb(decomp,"exp_medmgmt"),2]=sqrt(el(vartemp,1,1));

* The sum of all of the effects from above -- it's important that

```

```

        it be calculated this way, rather than just looking at the
        difference between integrated and non-integrated patients;
* It should be noted that the product below is algebraically
        equivalent to the sum of components (a)-(e) above;

* Make the second half of the predshare vector negative -- use
        this to generate the predicted value, but not the variance
        (because that was done in the patient share program);
mat predshare1=predshare[1..3,1];
mat predshare2=predshare[4..6,1];
scalar neg_one=-1;
mat predshare2_neg=predshare2*neg_one;
mat predsharenew=predshare1\predshare2_neg;

* OVERALL ;
* mat decomp[rownumb(decomp,"overall"),1]=
        preds_diff[rownumb(preds_diff,"inv_noninv"),1];
* mat decomp[rownumb(decomp,"overall"),2]=
        sqrt(preds_diff[rownumb(preds_diff,"inv_noninv"),1]);

mat predsnew=J(6,1,0);
mat predsnew[1,1]=preds[2..4,1];
mat predsnew[4,1]=preds[6..8,1];
mat variancenew=J(6,6,0);
mat variancenew[1,1]=variance[2..4,2..4];
mat variancenew[4,1]=variance[2..4,6..8];
mat variancenew[1,4]=variance[6..8,2..4];
mat variancenew[4,4]=variance[6..8,6..8];

mat decomp[1,1]=predsharenew'*predsnew[1...,1];
mat vartemp=predshare[1...,1]'\variancenew[1...,1...]*_
        predshare[1...,1]+
        predsnew[1...,1]'\Vshare*predsnew[1...,1];
mat decomp[1,2]=sqrt(el(vartemp,1,1));

mat list decomp;

* Export this matrix;
svmat double decomp;
outfile decompl-decomp2
        using decomp_outcome_`depvar'_`basic'_`int'.csv,
        comma wide replace;

* Drop all data and matrices from memory;
drop _all;

* End program;
end;

*****
* RUN OUR PROGRAMS
;
*****
* RUN OUTCOME REGRESSIONS ;
foreach var in hf365day ami365day dead365day postexpend

```

```
physclaims postexpend hospclaims postexpend { ;
decomp `var' noint basic;
decomp `var' int basic;
decomp `var' noint boot;
decomp `var' int boot;
};

display c(current_date);
display c(current_time);
clear;
exit;
```

F. SAS program to perform statistical tests in table 2

```

OPTIONS PS=32000 NOCENTER LS=256 NOLABEL NODATE NONNUMBER FORMDLIM=' '
MPRINT NOOVP ;

LIBNAME predict '/data';

DATA referral;
    SET predict.predictvars;
    IF zipcode = '' THEN DELETE;
run;

PROC UNIVARIATE DATA = referral NOPRINT;
    VAR area_invasivecard;
    OUTPUT OUT = percent pctlpts = 50 pctlpre = area_invasivecard_;
RUN;

DATA referral;
    IF _N_ = 1 THEN SET percent;
    SET referral;
* NOTE - THIS INDICATOR VARIABLE IS REVERSED TO MAKE TABLES THAT
SATISFY ABOVE INSTRUCTIONS ;
    area_invasive_hi = 1;
    IF area_invasivecard >= area_invasivecard_50 | area_invasivecard
=. THEN area_invasive_hi = 0;

    treatment = 4;
    IF revascproctype = 'ptca' AND phystype = 'Self-refer inv
intervention' THEN
        treatment = 1;
    IF revascproctype = 'ptca' AND phystype ~= 'Self-refer inv
intervention' THEN
        treatment = 2;
    IF revascproctype = 'cabg' THEN
        treatment = 3;
    allptca = 0;
    IF treatment = 1 | treatment = 2 THEN allptca = 1;

* NOTE - THIS INDICATOR VARIABLE IS REVERSED TO MAKE TABLES THAT
SATISFY ABOVE INSTRUCTIONS ;
    invasivecard_rev = 0;
    IF invasivecard = 0 THEN invasivecard_rev = 1;

RUN;

PROC TTEST DATA = referral;
    CLASS area_invasive_hi ;
    VAR invasivecard;
    TITLE 'Interventional';
PROC TTEST DATA = referral;
    CLASS area_invasive_hi ;
    VAR acuteprehospdaystot;
    TITLE 'Prior year acute days';
PROC TTEST DATA = referral;
    CLASS area_invasive_hi ;
    VAR acutehosppreexpendtot ;

```

```
TITLE 'Prior year acute expenditures';
PROC TTEST DATA = referral;
  CLASS area_invasive_hi ;
  VAR postexpend;
  TITLE 'Subsequent year expenditures';
RUN;
```