provider performance engine user's guide

Author: Seyoung Song

Date: 04 Sep 2018

Disclaimer: This document is intended for internal use only. Not to be disclosed.

1. data preparation

1-1. data loading and variable transformation

```
  options obs=max ;
  %put %sysfunc(getoption(work)) ;
  options extendobscounter=no compress=yes threads=yes fullstimer bufno=256 bufsize=128K;
  options minoperator mindelimiter=' ';
  options spool mprint symbolgen mlogic;
  libname medi  "U:\\Users\\Sy\\medi";
  libname medifmt  "U:\\Users\\Sy\\medi\\fmt";
  options fmtsearch=(WORK medifmt);

/****************inp****************/

%macro inp (input);
data inp_std_data0;
set medi.&input(keep=memid claimid enddate npi_op npi_hosp dischg proc1-proc10
procdt1-procdt10 cpt icd where=(year(enddate) in (2014, 2015)));
run;

/*proc sort  data=inp_std_data0 out=inp_std_data nodup; by _all_; run;*/
proc sort  data=inp_std_data0 out=inp_std_data nodupkey; by memid claimid; run;

/*============ICD9 Logic============*/

/*data inp_proc_2014_to_2015_L9;*/
data inp_proc_2014_to_2015;
format type $3. procid $6. proc_code $7. procdt mmddyy10.;
set inp_std_data;
type='inp';
array proc $ proc1 - proc10;
array dt $ procdt1 - procdt10;

obs=_n_;

do i=1 to dim(proc);
/*      if put(proc(i), $proc_L9_id.)^='' then do;*/
/*         procid = put(proc(i), $proc_L9_id.);*/
      if put(proc(i), $proc_id.)^='' then do;
         procid = put(proc(i), $proc_id.);
         var=cat(claimid," ", put(proc(i),4.)," is in");
         put var;
         proc_code=proc(i);
         procdt=dt(i);
         drop i;
         output;
      end;
end;
```

```
run;

%mend inp;

%inp (inp_std_data_2010_to_2015);

/*
%macro out(year=);
data out_proc_2014_to_2015;
format type $3.;
set medih.out_std_data_2010_to_2015(where=(year(enddate) in (2014, 2015)));
type='out';
    if put(cpt, $serv_id.)^='' then do;
       procid = put(cpt, $serv_id.);
       output;
    end;
run;
%mend out;
%out;*/


data inp_out_mem_2014_2015(keep=type year procid memid claimid startdate enddate npi_op npi_hosp dischg
set
/*inp_proc_2014_to_2015_L9(rename=(procdt=startdate))*/
/*inp_proc_2014_to_2015_L10*/
inp_proc_2014_to_2015(rename=(procdt=startdate));

/* out_proc_2014_to_2015;*/
year=year(enddate);
run;

/* name: mem */

proc sort data=inp_out_mem_2014_2015 out=medi.inp_out_mem_2014_2015 nodupkey;
by type year procid memid startdate; run;
```

- The medicare SAS table "inp_std_dat_2010_to_2015.sas7bdat" looks like below.

```
##   organizationid organizationname carrier     memid      claimid claimline
## 1              .               NA      NA 100002065 inp_2013_300         1
##    startdate    enddate PRVDR_NUM     provid   npi_hosp     npi_at
## 1 11/30/2013 12/04/2013    220080 1033421664 1033421664 1851623979
##       npi_op specialty   dx1   dx2   dx3  dx4  dx5   dx6  dx7   dx8   dx9
## 1 1356335327        NA 99672 42731 42822 4280 4111 41401 4019 V5861 V4582
##    dx10 proc1 proc2 proc3 proc4 proc5 proc6 proc7 proc8 proc9 proc10
## 1 28529  3606  3722    66  8853  8856    45    40    NA    NA     NA
##      procdt1    procdt2    procdt3    procdt4    procdt5    procdt6
## 1 12/03/2013 12/02/2013 12/03/2013 12/02/2013 12/02/2013 12/03/2013
##      procdt7 procdt8 procdt9 procdt10 cpt modifier drg rev billtype
## 1 12/03/2013       .       .        .           NA 249 120       11
##   claimtype sub_claimtype admissionsource admissiontype dischg  allowed
## 1 Inpatient            NA               1            NA      1 14008.06
##     billed     paid deductible coinsurance copay cob pos status network
## 1 48468.75 12824.06       1184           0     0   0  NA     NA      NA
```

```
##   paiddate firstname lastname gender dob icd
## 1        .        NA       NA     NA   .   9
```

- What does the first part in the macro do?

```
##        memid       claimid claimline    enddate     npi_op   npi_hosp
## 42 100027712 inp_2014_4251         1 11/15/2014 1760485460 1871606764
##    dischg cpt icd proc1 proc2 proc3 proc4 proc5 proc6 proc7 proc8 proc9
## 42      1       9  3606    66  3722  8853    45    40    NA    NA    NA
##    proc10    procdt1    procdt2    procdt3    procdt4    procdt5
## 42     NA 11/14/2014 11/14/2014 11/14/2014 11/14/2014 11/14/2014
##       procdt6 procdt7 procdt8 procdt9 procdt10
## 42 11/14/2014       .       .       .        .
```

It selects records having end(yeardate) = 2014 or 2015 and drops unused fields.

- what does "/=*ICD9 Logic*==/ do ?

```
##    proccode     memid       claimid claimline    enddate     npi_op
## 45     3606 100063900 inp_2014_9945         1 06/12/2014 1770571846
## 46     3606 100063900 inp_2014_9945         2 06/12/2014 1770571846
## 47     3606 100063900 inp_2014_9945         3 06/12/2014 1770571846
## 48     3606 100063900 inp_2014_9945         4 06/12/2014 1770571846
## 49     3606 100063900 inp_2014_9945         5 06/12/2014 1770571846
## 50     3606 100063900 inp_2014_9945         6 06/12/2014 1770571846
## 51     3606 100063900 inp_2014_9945         7 06/12/2014 1770571846
## 52     3606 100063900 inp_2014_9945         8 06/12/2014 1770571846
## 53     3606 100063900 inp_2014_9945         9 06/12/2014 1770571846
## 54     3606 100063900 inp_2014_9945        10 06/12/2014 1770571846
##      npi_hosp dischg cpt icd     procdt procid
## 45 1306938071      1       9 06/11/2014 130004
## 46 1306938071      1       9 06/11/2014 130004
## 47 1306938071      1       9 06/11/2014 130004
## 48 1306938071      1       9 06/11/2014 130004
## 49 1306938071      1       9 06/11/2014 130004
## 50 1306938071      1       9 06/11/2014 130004
## 51 1306938071      1       9 06/11/2014 130004
## 52 1306938071      1       9 06/11/2014 130004
## 53 1306938071      1       9 06/11/2014 130004
## 54 1306938071      1       9 06/11/2014 130004
```

It outputs through proc1-10 at each row the "proccode" contained in the ICD format, "procdt" and corresponding "procid" mapped from the format. Take a look at the first record with claimid "inp_2014_9945" and claimline 1. It has one row in the output because only the proccode 3606 matches the SAS format.

- how does ths SAS formats look like ?

```
##     start  end procid code_type code Category               code_desc
## 292  3606 3606 130004  icd9proc 3606       NA INS NONDRUG ELUT COR ST
##     Proc_type icd  label  fmtname type
## 292        NA   9 130004 $proc_id    C
```

This is "proc_910_format". It is used to map ICD code to procid.

```
##   start   end procid code_type  code Category                code_desc
## 1 other           NA                       NA
## 2 44950 44950 130001       cpt 44950       NA              APPENDECTOMY
## 3 44955 44955 130001       cpt 44955       NA      APPENDECTOMY ADD-ON
## 4 44960 44960 130001       cpt 44960       NA              APPENDECTOMY
```

```
## 5 44970 44970 130001       cpt 44970       NA LAPAROSCOPY APPENDECTOMY
##   Proc_type icd  label  fmtname type
## 1        NA  NA     NA $serv_id    C
## 2        NA  NA 130001 $serv_id    C
## 3        NA  NA 130001 $serv_id    C
## 4        NA  NA 130001 $serv_id    C
## 5        NA  NA 130001 $serv_id    C
```

This is "serv_format". It is used to map CPT code to procid. This is the format used to map outpatient data.

- What does the last part of macro do ? Let's take a look at the record with claimid "inp_2014_9945" and claimline 1.

```
##    proccode     memid       claimid claimline    enddate     npi_op
## 45     3606 100063900 inp_2014_9945         1 06/12/2014 1770571846
##      npi_hosp dischg cpt icd  startdate procid type year
## 45 1306938071      1       9 06/11/2014 130004  inp 2014
```

"proccdt" is renamed to "startdate". year(of enddate) and type (inp/out) are added new variables.

Finally, only "type year procid memid claimid startdate enddate npi_op npi_hosp dischg" are kept. They are sorted in the order of "type","year","procid","memid", "claimid".

```
##    type year procid     memid       claimid  startdate    enddate
## 45  inp 2014 130004 100063900 inp_2014_9945 06/11/2014 06/12/2014
##        npi_op   npi_hosp dischg
## 45 1770571846 1306938071      1
```

- The table in this layout is the master table (named with adjunct "*mem*").

1-2. variable generation

1. length of stay

```
proc sql;
create table procedure_all as
select * from medi.inp_out_mem_2014_2015
where (year=2015 or year=2014)
and npi_op in
(select distinct npi_op from medi.inp_out_mem_2014_2015 where year=2014 intersect
select distinct npi_op from medi.inp_out_mem_2014_2015 where year=2015); quit;

data procedure_all(replace=yes);
set procedure_all;
los = enddate-startdate;
if enddate<=startdate then los=1;
run;
```

- What does this part do ? los is calculated as

```
los = enddate-startdate;
if enddate<=startdate then los=1;
```

and los is added to the master table as below.

```
##     type year procid     memid       claimid  startdate    enddate
## 466  inp 2014 130004 100027712 inp_2014_4251 2014-11-14 2014-11-15
##         npi_op   npi_hosp dischg los
## 466 1760485460 1871606764      1  NA
```

```
##    type year procid memid claimid startdate enddate npi_op npi_hosp dischg
## NA <NA> <NA>     NA    NA    <NA>      <NA>    <NA>     NA       NA     NA
##    los
## NA  NA
```

```
## [1] NA
```

2. readmission

```
data inp_2014_2015;
set medi.inp_std_data_2010_to_2015
(rename=(dx1=dx_1 dx2=dx_2 dx3=dx_3 dx4=dx_4 dx5=dx_5
                                        dx6=dx_6 dx7=dx_7 dx8=dx_8 dx9=dx_9 dx10=dx_10));
where=(year(enddate) in (2014, 2015));

array dx_o dx_1-dx_10;
array dx_n $15 dx1-dx10;
array proc proc1-proc6 cpt;

* change dx_ format to icd 9 or 10 + dx;
do i=1 to dim(dx_n);
*  icd: ind for  ICD 9 OR 10;
if dx_o(i) ^='' then dx_n(i) =cats('icd',icd,'_',upcase(dx_o(i)));

end;
do j=1 to dim(proc);

 if proc(j) ^='' then proc(j) =upcase(proc(j));
end;
drop i j dx_1-dx_10;
run;
```

```
##     organizationid organizationname carrier     memid       claimid
## 118              .               NA      NA 100063900 inp_2014_9945
##     claimline  startdate    enddate PRVDR_NUM     provid   npi_hosp
## 118         1 06/10/2014 06/12/2014    100007 1306938071 1306938071
##         npi_at     npi_op specialty       dx1      dx2      dx3       dx4
## 118 1699708032 1770571846        NA icd_99672 icd_5854 icd_3572 icd_41401
##           dx5       dx6      dx7      dx8      dx9 dx10 proc1 proc2 proc3
## 118 icd_25060 icd_40390 icd_2724 icd_V103 icd_7245       3606    66    45
##     proc4 proc5 proc6 proc7 proc8 proc9 proc10    procdt1    procdt2
## 118    40  <NA>  <NA>  <NA>  <NA>  <NA>   <NA> 06/11/2014 06/11/2014
##        procdt3    procdt4 procdt5 procdt6 procdt7 procdt8 procdt9 procdt10
## 118 06/11/2014 06/11/2014       .       .       .       .       .        .
##     cpt modifier drg rev billtype claimtype sub_claimtype admissionsource
## 118          NA 249 206       11 Inpatient            NA               1
##     admissiontype dischg  allowed   billed     paid deductible coinsurance
## 118            NA      1 11095.82 51742.86 11095.82          0           0
##     copay cob pos status network paiddate firstname lastname gender dob
```

```
## 118     0   0  NA     NA      NA        .        NA       NA     NA    .
##     icd
## 118   9
```

- What does this part do ? It reformats dx to have "icd_" prefix and capitalize proc. This is to filter out certain claims with dxs and procs using the predefined formats.

```
libname ecci v9 "O:\\SAS\\Formats\\CCI";
libname fmtv4 v9 "O:\\SAS\\formats\\eh\\V4";
option fmtsearch = (eh);
options fmtsearch=(WORK CCI FMTV4);
libname hq2 odbc dsn="DB_HQ2" schema="dbo";

%let rstart=%sysfunc(mdy(1,1,2014));
%let rend=%sysfunc(mdy(12,31,2015));
%let client=medicare;
%let provider1 = npi_op;


%macro Readmission_npi(client=);

/* why exclueded*/
data &client._plan_readmit;
set inp_2014_2015;
array dx $ dx1-dx10;
array proc $ proc1-proc6;

do i=1 to dim(dx);

if put(dx(i),$Csection_v4_dx.)='in' or put(dx(i),$chemo_v4_dx.)='in' or
/*$Csection_v4_dx. format for iop?*/

put(dx(i),$rehab_v4_dx.)='in' then output;
end;


do j=1 to dim(proc);
if put(proc(j),$Marrowtrans_v4_proc.)='in' or put(proc(j),$Organtrans_v4_proc.)='in' or
put(proc(j),$Csection_v4_proc.)='in'
or put(proc(j),$chemo_v4_proc.)='in' then output;
end;
run;


data &client._expired_clm(keep=claimid);
set medi.inp_out_mem_2014_2015;
if dischg in  ('20','40','41','42') then output;
run;
*-------------------------------------------------------------;

proc sql;
create table &client._medical_readmit_i as
select  *
from medi.inp_out_mem_2014_2015
where  &rstart<=startdate<=&rend and claimid not in  (select claimid from &client._expired_clm)
```

```
;
quit;
```

- What does this part do ?

Table with suffix _i filters out expired claims with dischg code in '20','40','41','42'. And claims with start and end dates between 2014 and 2015 are filtered in.

```
readmit_i
```

```
##     type year procid     memid       claimid  startdate    enddate
## 466  inp 2014 130004 100027712 inp_2014_4251 2014-11-14 2014-11-15
##         npi_op   npi_hosp dischg los
## 466 1760485460 1871606764      1  NA
```

```
proc sql;
create table &client._medical_readmit_r as
select *
from inp_2014_2015
where &rstart<=startdate<=&rend and substr(billtype,1,2)='11'
      and claimid not in (select distinct claimid from &client._plan_readmit where claimid^='')

;
quit;
```

As mentioned previously, table with suffix _r filters out claims associated with certain dxs and procs. Also, only inpatient claims are filtered in.

```
readmit_r
```

```
##     organizationid organizationname carrier     memid       claimid
## 118              .               NA      NA 100063900 inp_2014_9945
##     claimline  startdate    enddate PRVDR_NUM     provid   npi_hosp
## 118         1 06/10/2014 06/12/2014    100007 1306938071 1306938071
##         npi_at     npi_op specialty       dx1      dx2      dx3       dx4
## 118 1699708032 1770571846        NA icd_99672 icd_5854 icd_3572 icd_41401
##           dx5       dx6      dx7      dx8      dx9 dx10 proc1 proc2 proc3
## 118 icd_25060 icd_40390 icd_2724 icd_V103 icd_7245       3606    66    45
##     proc4 proc5 proc6 proc7 proc8 proc9 proc10    procdt1    procdt2
## 118    40  <NA>  <NA>  <NA>  <NA>  <NA>   <NA> 06/11/2014 06/11/2014
##        procdt3    procdt4 procdt5 procdt6 procdt7 procdt8 procdt9 procdt10
## 118 06/11/2014 06/11/2014       .       .       .       .       .        .
##     cpt modifier drg rev billtype claimtype sub_claimtype admissionsource
## 118          NA 249 206       11 Inpatient            NA               1
##     admissiontype dischg  allowed   billed     paid deductible coinsurance
## 118            NA      1 11095.82 51742.86 11095.82          0           0
##     copay cob pos status network paiddate firstname lastname gender dob
## 118     0   0  NA     NA      NA        .        NA       NA     NA   .
##     icd
## 118   9
```

```
proc sort data=&client._medical_readmit_i (keep=memid startdate enddate &provider1. procid) nodupkey;
by memid startdate enddate;run;
proc sort data=&client._medical_readmit_r (keep=memid startdate enddate &provider1. ) nodupkey;
by memid startdate enddate;run;

data &client._medical_readmit_i(replace=yes);
```

```
set &client._medical_readmit_i;
where startdate^=. and enddate^=. and startdate<=enddate;
run;

data &client._medical_readmit_r(replace=yes);
set &client._medical_readmit_r;
where startdate^=. and enddate^=. and startdate<=enddate;
run;
```

After this, only claims with startdate<=enddate are filtered in. Only some fields are kept for both tables.

```
readmit_i
```

```
##         memid  startdate    enddate     npi_op procid
## 466 100027712 2014-11-14 2014-11-15 1760485460 130004
```

```
readmit_r
```

```
##         memid  startdate    enddate     npi_op
## 118 100063900 06/10/2014 06/12/2014 1770571846
```

Now, ipstart and ipend are set from the start and end dates. See the logics below.

```
data &client._medical_readmit_i(replace=yes);
    set &client._medical_readmit_i;
    by memid;

    retain ipstart ipend; *retain firs claim ipstart ipend  ;

    if first.memid then do;
        ipstart=startdate;
        ipend=enddate;
    end;
    else do;
        if startdate-ipend<=1 then do;
            startdate=ipstart;
            ipend=enddate;
        end;
        else do;
            ipstart=startdate;
            ipend=enddate;
        end;
    end;
    format ipstart ipend mmddyy10.;
/*  keep memid ipstart ipend ;*/
    keep memid ipstart ipend startdate enddate &provider1. procid;
run;
```

```
readmit_i
```

```
##         memid  startdate    enddate     npi_op procid    ipstart
## 466 100027712 2014-11-14 2014-11-15 1760485460 130004 2014-11-14
##          ipend
## 466 2014-11-15
```

```
data &client._medical_readmit_r;
    set &client._medical_readmit_r;
    by memid; *by memid;
```

```
    retain ipstart ipend;   *retain firs claim ipstart ipend  ;
    if first.memid then do;

        ipstart=startdate;
        ipend=enddate;
    end;

    else do;
        if ipstart<=startdate<=ipend+1 then do;

            startdate=ipstart;
            ipend=max(enddate,ipend); /*why different ?*/
        end;
        else do;
            ipstart=startdate;
            ipend=enddate;
        end;
    end;
    format ipstart ipend mmddyy10.;
/*  keep memid ipstart ipend ;*/
    keep memid ipstart ipend startdate enddate &provider1.;
run;
```

Same is processed for _r table except the difference that ipend=max(enddate,ipend).

```
readmit_r
```

```
##         memid  startdate    enddate     npi_op    ipstart      ipend
## 118 100063900 06/10/2014 06/12/2014 1770571846 06/10/2014 06/12/2014
```

```
proc sort data=&client._medical_readmit_i nodupkey;by memid ipstart descending ipend;run;
proc sort data=&client._medical_readmit_r nodupkey;by memid ipstart descending ipend;run;

/*
proc sort data=&client._medical_readmit_i nodupkey;
by memid ipstart;
run;
proc sort data=&client._medical_readmit_r out=&client._medical_readmit_r4 nodupkey;
by memid ipstart;
run;
*/
```

After this, _i and _r table look like

```
readmit_i
```

```
##         memid  startdate    enddate     npi_op procid    ipstart
## 466 100027712 2014-11-14 2014-11-15 1760485460 130004 2014-11-14
##          ipend
## 466 2014-11-15
```

```
readmit_r
```

```
##         memid  startdate    enddate     npi_op    ipstart      ipend
## 118 100063900 06/10/2014 06/12/2014 1770571846 06/10/2014 06/12/2014
```

** strange.. take a look **

```
#dat[order(dat$memid),c("memid","startdate")][1:50,]
```

Why it is made into two separate tables by the way ? To filter in the readmission matching the condition (with in 30 days). The _i table provides the inpatient end dates from which the gap to the next readmission start date from the _r table is calculated. See the logics below.

```
/* join i an r */

proc sql;
create table &client._index_readmit as
select distinct a.memid,a.ipstart,a.ipend,
                b.ipstart as r_ipstart, b.ipend as r_ipend
               ,a.&provider1.
               ,a.procid

from &client._medical_readmit_i a left join &client._medical_readmit_r b
on a.memid=b.memid and 1<b.ipstart-a.ipend<=30   /*b ip , a ip and op */
order by a.memid,a.ipstart,a.ipend
;
quit;


proc sort data=&client._index_readmit nodupkey;
by memid ipstart ipend r_ipstart ;
run;
```

After it runs, r_ipstart and r_ipend are added. NA means no readmission record for a particular member ipstart ipend combination matching the readmission criteria above.

```
##       memid ipstart ipend r_ipstart r_ipend     npi_op procid
## 1 100027712    <NA>  <NA>        NA      NA 1760485460 130004
## 2 100029817    <NA>  <NA>        NA      NA 1154493070 130004
## 3 100050843    <NA>  <NA>        NA      NA 1285651109 130004
## 4 100053415    <NA>  <NA>        NA      NA 1992739668 130004
## 5 100063900    <NA>  <NA>        NA      NA 1770571846 130004
```

```
* inp start and end date before 20151130

data &client._index_readmit;
set &client._index_readmit;
where &rstart<=ipstart<=intnx('month',&rend,-1,'e')
and &rstart<=ipend<=intnx('month',&rend,-1,'e');
run;

proc sort data=&client._index_readmit;
by memid  r_ipstart descending ipstart;
run;


data &client._index_readmit;
set &client._index_readmit;

by memid r_ipstart;
```

indicator is added to filter out 2+ readmission in calculation.

```
*readmission ind :
*ind 0 : first claim or no readmission startdate
*ind 1 : if first readmissino
*ind 2 : 2+ readmission
* if ind 2, rmv r_ipstart   r_ipend ;

if first.memid then ind=0;

if r_ipstart=. then ind=0;
else if first.r_ipstart then ind=1;
else ind=2;

if ind=2 then do;
r_ipstart=.;
r_ipend=.;

end;
run;
```

```
##      memid ipstart ipend r_ipstart r_ipend npi_op procid ind
## NA      NA    <NA>  <NA>        NA      NA     NA     NA   0
## NA.1    NA    <NA>  <NA>        NA      NA     NA     NA   0
## NA.2    NA    <NA>  <NA>        NA      NA     NA     NA   0
## NA.3    NA    <NA>  <NA>        NA      NA     NA     NA   0
## NA.4    NA    <NA>  <NA>        NA      NA     NA     NA   0
```

Finally, readmission rate is calculated by provider and procedure. See the logics below.

```
proc sql;
create table &client._readmit_rate as
/*select distinct &provider1._1,&provider2._1,procid*/
select distinct &provider1.,procid       ,count(*) as admission ,
    sum( r_ipstart^=.) as readmission,
sum( r_ipstart^=.)/count(*) as readmission_rate

from &client._index_readmit

/*where ^missing(&provider1._1) and ^missing(&provider2._1)*/
group by &provider1.,procid
order by procid,&provider1.,admission desc
;
quit;

%mend Readmission_npi;

%Readmission_npi(client=cms);
```

```
##      npi_op procid admission readmission readmission_rate
## 1        NA     NA       237           0                0
## NA       NA     NA        NA          NA               NA
## NA.1     NA     NA        NA          NA               NA
## NA.2     NA     NA        NA          NA               NA
## NA.3     NA     NA        NA          NA               NA
## NA.4     NA     NA        NA          NA               NA
## NA.5     NA     NA        NA          NA               NA
```

```
## NA.6     NA     NA        NA          NA               NA
## NA.7     NA     NA        NA          NA               NA
## NA.8     NA     NA        NA          NA               NA
```

3. cci

```
data  medi.inp_out_std_2013_2015;
set
medi.inp_std_data_2010_to_2015(in=a keep=memid startdate enddate dx1-dx10 icd dischg npi_op
where=(year(enddate) in (2013, 2014, 2015)))
;

/*medih.out_std_data_2010_to_2015(in=b keep=memid startdate enddate dx1-dx10 icd dischg npi_op where=( *


if a then type='inp';
if b then type='out';
run;


proc sql;
create table medical_cci as
select
 b.procid
,b.memid
/* cci index*/
,b.startdate as index_startdate format=mmddyy10.
,b.enddate as index_enddate format=mmddyy10.
,b.npi_op

,a.startdate,a.enddate
,a.dx1,a.dx2,a.dx3
,a.dx4,a.dx5,a.dx6
,a.dx7,a.dx8,a.dx9,a.dx10
,a.icd

from  medi.inp_out_std_2013_2015  a,
medi.inp_out_mem_2014_2015  b                        /* b is used as index*/


where a.memid=b.memid /************ why join with only member id ?*************/

/* claim within last one year? */
and intnx('year',b.startdate,-1,'sameday') < a.startdate <= b.startdate
;

quit;
```

- What does this part do ?

It creates a table that contains information about dxs to be used for CCI with claims ending in 2013-2015. While the CCI index start and end dates are in 2014-2015, dxs to be used for CCI scoring come from another claim of the same member occured within a year before the index claim. This is because the CCI was originally developed to predict 1-year mortality.

```
data medical_cci(replace=yes);
set medical_cci
(rename=(dx1=dx_1 dx2=dx_2 dx3=dx_3 dx4=dx_4 dx5=dx_5
                                        dx6=dx_6 dx7=dx_7 dx8=dx_8 dx9=dx_9 dx10=dx_10));
array dx_o dx_1-dx_10;
array dx_n $15 dx1-dx10;

do i=1 to dim(dx_n);
                    if dx_o(i) ^='' then dx_n(i) =cats('icd',icd,'_',upcase(dx_o(i)));
end;
drop i dx_1-dx_10;
run;

/*proc sort data=medical_cci;by procid memid index_startdate index_enddate;run;*/
```

As before, 'dx' is recoded to use the SAS format.

```
libname ehcci v9 "O:\SAS\Formats\CCI";

data claims;
set medical_cci;

array dx $ dx1-dx10;

do i=1 to dim(dx);

if put(dx(i), $mi_dx_cci.)='in' then mi=1;
if put(dx(i), $chf_dx_cci.)='in' then chf=1;
if put(dx(i), $pvd_dx_cci.)='in' then pvd=1;
if put(dx(i), $cvd_dx_cci.)='in' then cvd=1;
if put(dx(i), $ccidem_dx_cci.)='in' then ccidem=1;
if put(dx(i), $cpd_dx_cci.)='in' then cpd=1;
if put(dx(i), $rhemz_dx_cci.)='in' then rhemz=1;
if put(dx(i), $pud_dx_cci.)='in' then pud=1;
if put(dx(i), $mliverd_dx_cci.)='in' then mliverd=1;
if put(dx(i), $diabnc_dx_cci.)='in' then diabnc=1;
if put(dx(i), $diabc_dx_cci.)='in' then diabc=1;
if put(dx(i), $hpplegia_dx_cci.)='in' then hpplegia=1;
if put(dx(i), $renald_dx_cci.)='in' then renald=1;
if put(dx(i), $cancer_dx_cci.)='in' then cancer=1;
if put(dx(i), $msliver_dx_cci.)='in' then msliver=1;
if put(dx(i), $mcancer_dx_cci.)='in' then mcancer=1;
if put(dx(i), $aids_dx_cci.)='in' then aids=1;

end;
/*
do j=1 to dim(proc);
if put(proc(j), $pvd_proc_cci.)='in' then pvd=1;
end;*/

keep dx1-dx10 mi chf pvd cvd ccidem cpd rhemz pud mliverd
diabnc diabc hpplegia renald cancer msliver mcancer aids
memid procid index_startdate npi_op
;
```

```
run;
proc sort data=claims; by memid procid index_startdate npi_op;
run;

proc means data=claims sum noprint;
var mi chf pvd cvd ccidem cpd rhemz pud mliverd
diabnc diabc hpplegia renald cancer msliver mcancer aids;

by memid procid index_startdate npi_op; *sum by memid procid index_startdate npi_op;

output out=sumcci
sum=smi schf spvd scvd sccidem scpd srhemz spud smliverd
sdiabnc sdiabc shpplega srenald scancer smsliver smcancer
saids;
run;
```

and it counts and sums each dx counted in CCI per member, procedure, index startdate, provider.

Now, CCI scores are calculated by adding scores for meeting each condition in each scoring group.

```
data cci;
set sumcci; by memid procid index_startdate npi_op;
array s smi schf spvd scvd sccidem scpd srhemz spud smliverd
sdiabnc sdiabc shpplega srenald scancer smsliver smcancer
saids;
array i imi ichf ipvd icvd iccidem icpd irhemz ipud imliverd
idiabnc idiabc ihpplega irenald icancer imsliver imcancer
iaids;
do over i;
if s>=1 then i=1;
else i=0;
end;
cci1=0; cci2=0; cci3=0; cci6=0;

array one imi ichf ipvd icvd iccidem icpd irhemz ipud imliverd idiabnc;
array two idiabc ihpplega irenald icancer;
array six imcancer iaids;

do over one;
if one=1 then cci1=cci1+1;
end;
do over two;
if two=1 then cci2=cci2+2;
end;
do over six;
if six=1 then cci6=cci6+6;
end;
if imsliver=1 then cci3=cci3+3;
cci=cci1+cci2+cci3+cci6;
run;

proc sort data=cci nodupkey;
by memid procid index_startdate npi_op;
run;
```

4. age

Member age information comes from a separate source. This seems to be a huge table.

```
libname member "M:\medicare_sas_data\Source Files\data";

proc sql;
create table age as
select 2013 as year, DESY_SORT_KEY as memid , a.* from member.Dnmntr_saf_lds_2013 a
where age ^=. and DESY_SORT_KEY ^= '000000000'
union
select 2014 as year, DESY_SORT_KEY as memid, a.* from member.Dnmntr_saf_lds_2014 a
where age ^=. and DESY_SORT_KEY ^= '000000000'
union
select 2015 as year, DESY_SORT_KEY as memid, a.* from member.Dnmntr_saf_lds_2015 a
where age ^=. and DESY_SORT_KEY ^= '000000000';
quit;

proc sort data=age nodupkey;
by year memid; run;
```

The last step is the left join of the master table with the additional tables generated for cci, age, readmission on provider and procedure and to summarize the key measures (los,age, cci, volume, inpatient volume) by provider and procedure.

```
proc sql;
create table details as
select a.*,
b.cci,
c.age,

(case when
type='inp' then 1 else 0 end) as inp

from procedure_all a left join cci b
on a.procid=b.procid
and a.memid=b.memid
and a.startdate=b.index_startdate
and a.npi_op=b.npi_op

left join age c
on a.memid=c.memid
and a.year=c.year
;
quit;


proc sql;
create table summary0 as
select distinct   /** not needed */
npi_op, procid

,count(*) as volume           /*** volume = # clm by provider and proc */
,sum(inp) as inp              /*** inp = # volume with ind ip **/
```

```
,avg(los) as avg_los
    ,avg(cci) as avg_cci
            ,avg(age) as avg_age

from details
group by npi_op,procid
order by npi_op,procid
;
quit;
proc sql;


create table summary as
select
 a.npi_op
,a.procid
,a.volume
,b.admission
,(a.volume - b.admission) as diff

,a.inp
,a.avg_los
,a.avg_cci
,a.avg_age

,b.readmission
,b.readmission_rate

from summary0 a
left join cms_readmit_rate b   /*grouped by proc and npi */
on a.npi_op=b.npi_op and a.procid=b.procid
;
quit;


/* remove missing vol, cci, age, los, npi(npi_op as npi) ,
readmission and admission */
proc sql;
create table summary as
select
 npi_op as npi
,procid
,volume
,inp
,case when missing(readmission_rate) then 0
      else readmission_rate
      end as readmission_rate
,avg_los
,avg_cci
,avg_age
,case when missing(admission) then 0
      else admission
      end as admission
```

```
from summary

where ^missing(volume) and  ^missing(avg_cci) and  ^missing(avg_age)
 and avg_los < 500 and  ^missing(npi)

order by npi_op,procid
;
quit;


data medi.summary;
set summary;
if procid in ('130006','130007') and volume ^= inp then delete;
run;
```

The data in the layout below is called "summary" table. This is the final output to be used for the remaining parts.

```
##       provid procid vol       los readmit   avgage   avgcci inp admission
## 1 1003022591 130011   3  9.666667       0 68.66667 1.666667   3         2
## 2 1003843921 130009   3  1.666667       0 50.33333 1.000000   3         3
## 3 1003851817 130011   3 11.000000       0 81.66667 1.000000   3         1
## 4 1003880295 130004   3  3.000000       0 67.00000 2.333333   3         3
## 5 1003888520 130004   3  2.333333       0 71.00000 2.333333   3         3
```

The following are to validate and further cleanse the final output. It does not create additional inputs.

1-3. cross check against other data repositories

It is to check the volumes. It gets doctor specialities by joining with the HCE+ database on provider id. This is to remove the records with procedures that doesn't match with the provider's specialties. After joining with HCE databaes on npi_op, the records with volume less than 3 are filtered out.

2. adjusting primary outcomes by risk characteristics

1. los

The distribution of los as below. A lot of them are 1 .

```
##
##      1      2      3      4      5      6      7      8      9     10
## 166233  52063  37581  16779   7706   6225   4501   3704   2403   1878
##     11     12     13     14     15     16     17     18     19     20
##   1217    999    556    507    328    303    161    176    103     83
##     21     22     23     24     25     26     27     28     29     30
##     64     60     49     39     31     32     11     15      9     11
##     31     32     33     34     35     36     37     38     39     40
##     10     17      9     10      7      7      7      3      5      2
##     41     42     43     44     46     47     48     49     50     51
##      2      5      1      2      3      3      1      3      1      1
##     53     56     67     78     80    134    225
##      1      1      1      1      1      1      1
```

This means los is left censored at 1. Los is 1 if enddate=startdate (no case such that enddate < startdate which is a coding error.) Assuming los less than a day cannot be recorded, observable los 1 as below.

$$y_i = \begin{cases} y_i^* & \text{if } y_i^* > 1 \\ 1 & \text{if } y_i^* <= 1 \end{cases}$$

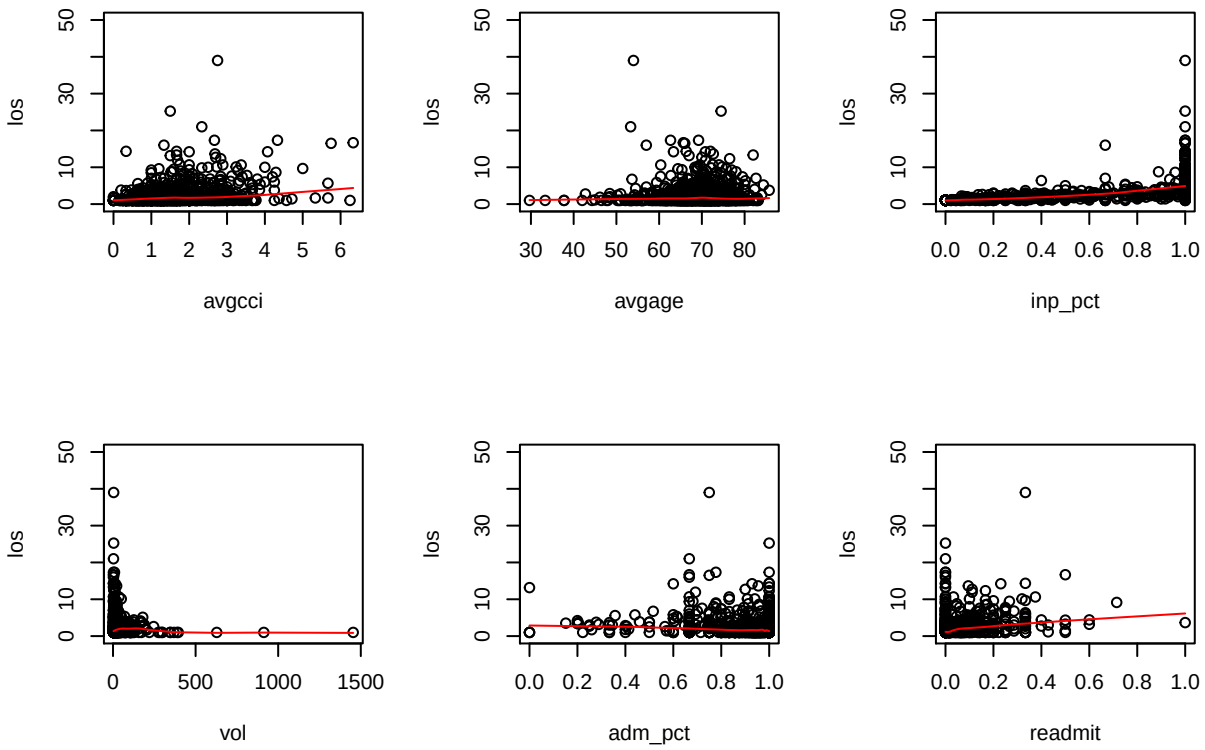
where y_i^* is actual los (and max(1,y_i^*) is what is observed).

A censored regression model was used to estimate linear relationships between actual los and independent variables. In addition, a parametric disturbance term was assumed to capture random influences on this relationship. For example,

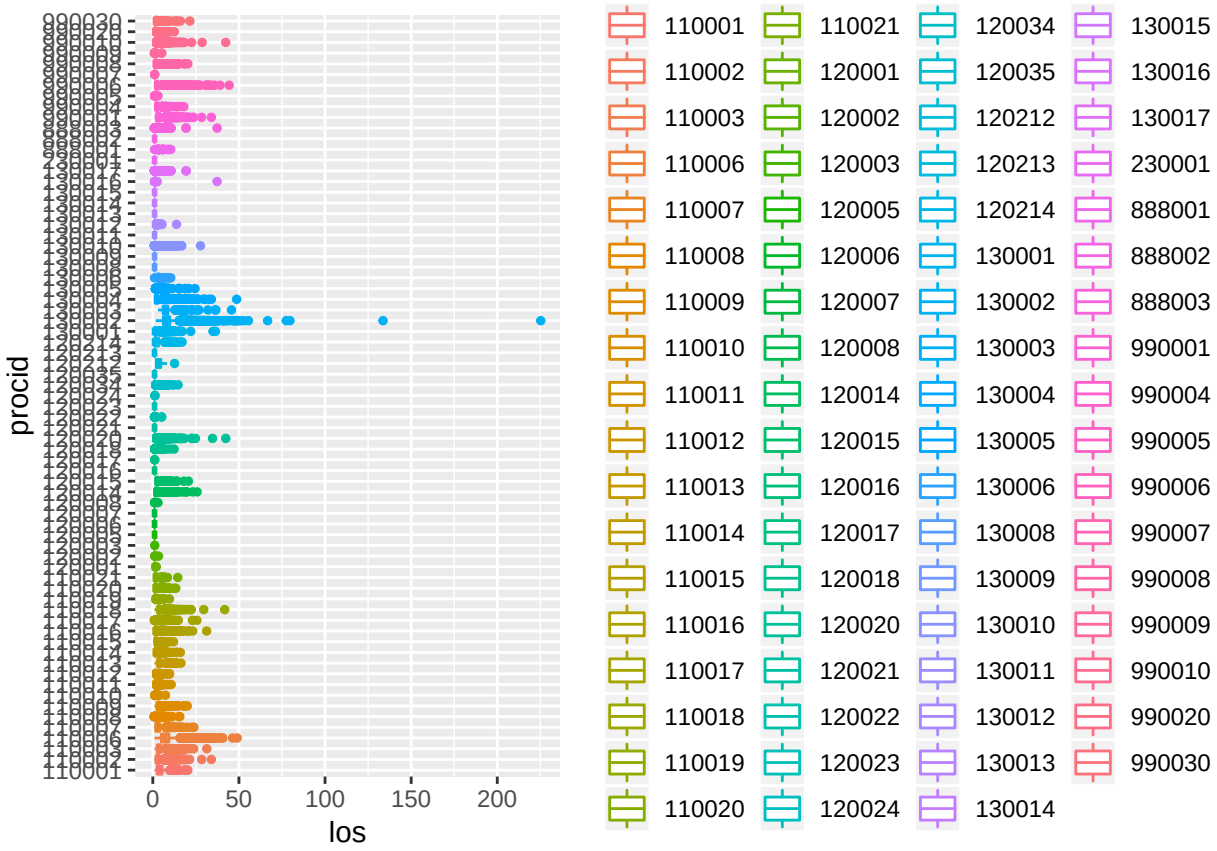
$y^* = x\beta + \mu,\ \mu|x \sim logistic(0, s)$

$y^* = x\beta + \mu,\ \mu|x \sim N(0, \sigma^2)$

What are the risk characteristics that seem most closely related to los ?



How about procedure ? Are there more or less risky procedures in terms of los out of these 71 procedures ?



The fitted model is as below.

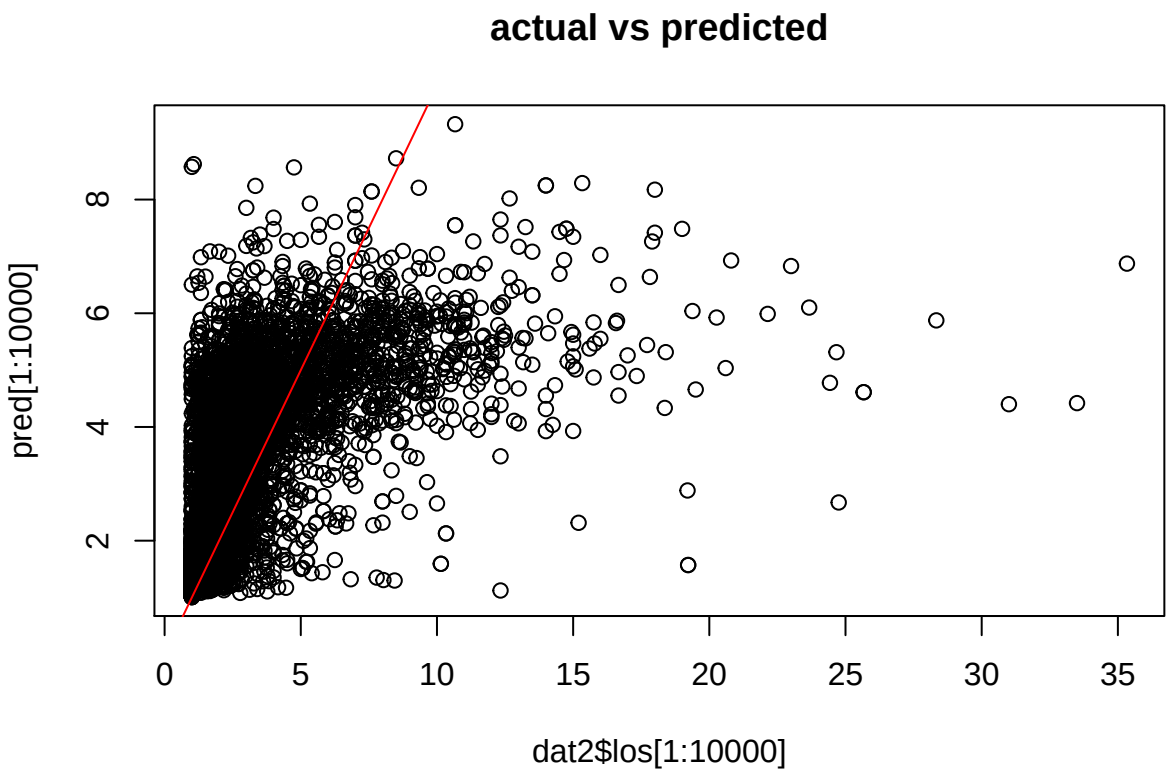
```
##
## Call:
## survreg(formula = Surv(los, los > 1, type = "left") ~ inp_pct +
##     avgcci + avgage + readmit, data = dat2, dist = "gaussian")
##                Value Std. Error     z p
## (Intercept) -4.7980   0.055603 -86.3 0
## inp_pct      6.0264   0.013801 436.7 0
## avgcci       0.5019   0.005381  93.3 0
## avgage       0.0329   0.000797  41.3 0
## readmit      3.8816   0.049856  77.9 0
## Log(scale)   0.8635   0.001687 511.7 0
##
## Scale= 2.37
##
## Gaussian distribution
## Loglik(model)= -431669.4   Loglik(intercept only)= -559490.6
##  Chisq= 255642.4 on 4 degrees of freedom, p= 0
## Number of Newton-Raphson Iterations: 6
## n= 303933
```

Now comparing predicted los $\hat{Y}_i = (1 - \Phi(\frac{1-x_i^{'}\hat{\beta}}{\hat{\sigma}})) * (x_i^{'}\hat{\beta} + \hat{\sigma} * \lambda_i),\ \lambda_i = \frac{\phi(x_i^{'}\hat{\beta}/\hat{\sigma})}{\Phi(x_i^{'}\hat{\beta}/\hat{\sigma})}$

$\hat{Y}_i = \hat{\sigma} * \ln(1 + exp(x_i^{'}\hat{\beta} - 1)/\hat{\sigma}))$ (if logistic) vs actual,

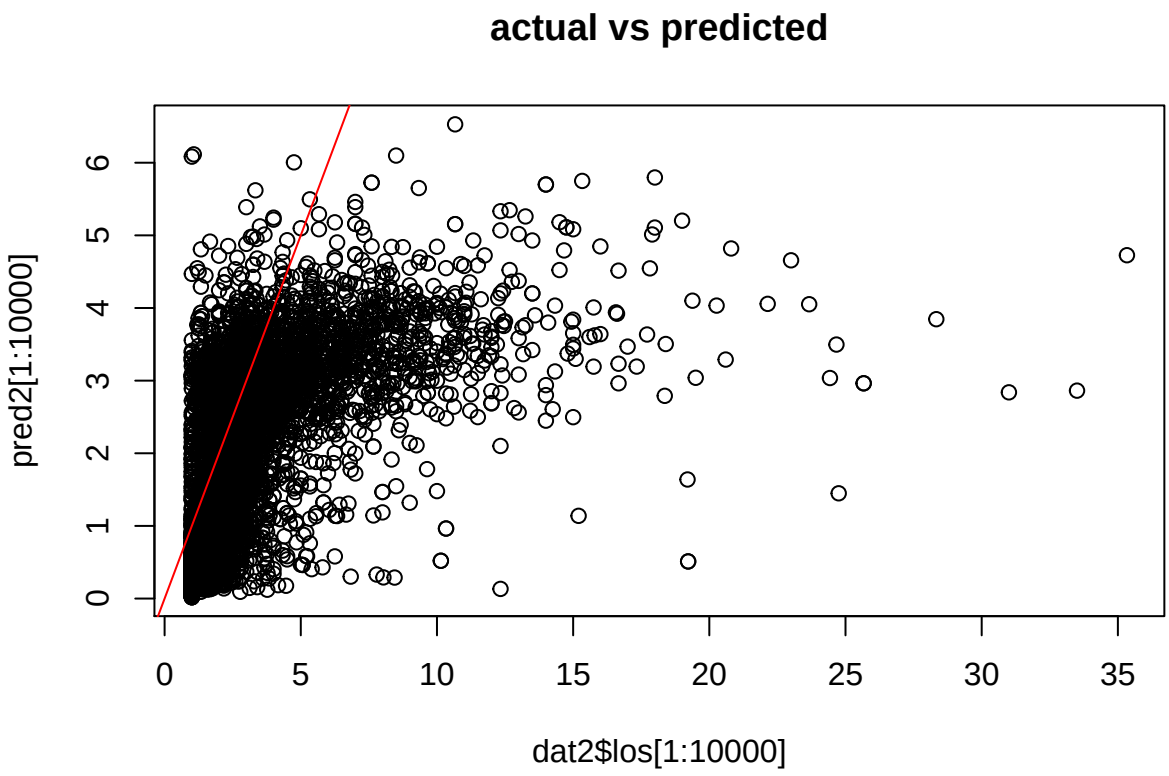
```
## [1] "r^2"
```

```
## Error in cor(fitted(m), dat2$los): 'x' must be numeric
```



Most actual loss are located right to the 45-degree line. The predicted range is much less than actual range, which may imply model misspecification. Let's see how the result changes if using logistic distribution.

```
##
## Call:
## survreg(formula = Surv(los, los > 1, type = "left") ~ inp_pct +
##     avgcci + avgage + readmit, data = dat2, dist = "logistic")
##               Value Std. Error     z        p
## (Intercept) -3.0932   0.039848 -77.6 0.00e+00
## inp_pct      4.5121   0.010063 448.4 0.00e+00
## avgcci       0.3485   0.003957  88.1 0.00e+00
## avgage       0.0261   0.000566  46.1 0.00e+00
## readmit      3.0276   0.039148  77.3 0.00e+00
## Log(scale)  -0.0224   0.002105 -10.6 2.11e-26
##
## Scale= 0.978
##
## Logistic distribution
## Loglik(model)= -396049.5   Loglik(intercept only)= -535472.9
##  Chisq= 278846.8 on 4 degrees of freedom, p= 0
## Number of Newton-Raphson Iterations: 5
## n= 303933
```




```
## [1] 863350.8
```

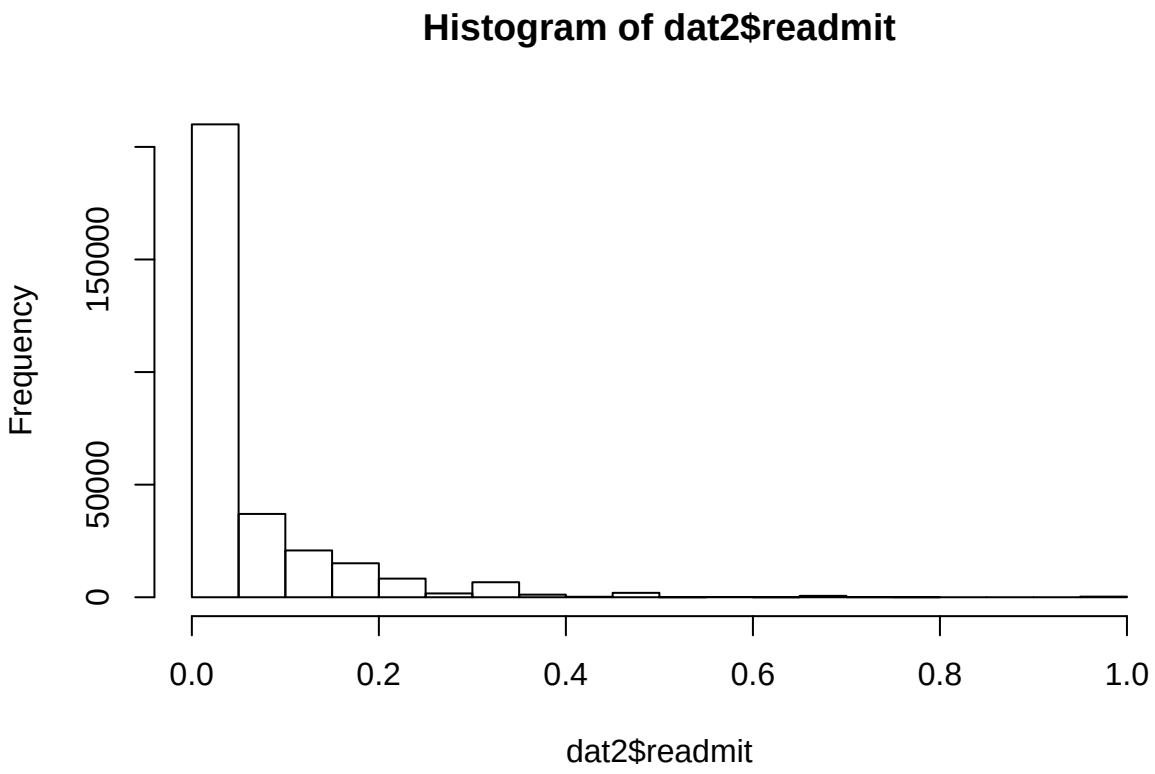
```
## [1] 792111
```

The results don't change a lot. It has less to do with the assumption about error distribution.

2. readmission rate

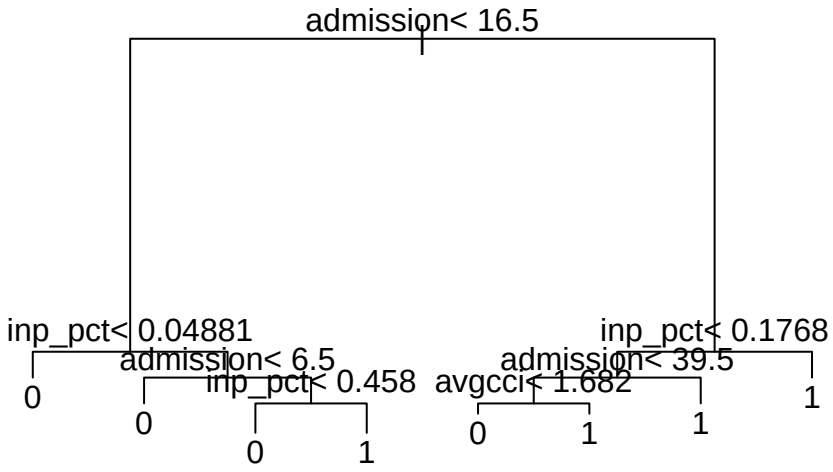
A lot of data points have readmission zero.

```
## [1] 0.5838524
```



They set an indicator variable (1 if readmission >0 and 0 otherwise) and tried to predict this indicator instead of readmission rate.

and they used classification tree, random forest, svm for this prediction. Let's do decision tree for example. Following is a fitted classification tree that is basically the same.




```
##  admission    inp_pct     avgcci     avgage    adm_pct
## 35172.3942 10968.6778  2072.5575  1043.7778   364.2775
```

and prediction on this readmission indicator as below

```
##        actual
## predict      0      1
##       0 151208  44965
##       1  26244  81516
```

```
## [1] "accuracy"
```

```
## [1] 0.7657082
```

and did the same for the remaining two models and compared them for the best accuracy and finally took the prediction results from randomforest as below.

```
##       provid procid vol      los readmit inp admission readmit_ref avgcci
## 1 1003000423 990008   3 2.333333    0.00   2         2           0   1.00
## 2 1003000480 120014   4 1.250000    0.25   4         4           1   1.25
##     avgage   inp_pct   adm_pct predictRF1 predictRF1_prob
## 1 44.66667 0.6666667 0.6666667          0           0.250
## 2 76.00000 1.0000000 1.0000000          0           0.082
```

predictRF1 is the prediction of class (0 of 1). predictRF1_prob is the prediction of probability of assigned to that class." readmit is the readmission rate and readmit_ref is the indicator used for prediction.

Using these predicted readmission rates from RF, readmision rates greater than 0 are predicted again from a normal regression with log link fitted on the readmission rate.

In other words, Readmission rate predicted = Pr (readmission rate > 0) * Pr (predicted readmission rate | readmission rate >0)

(*Details will come later.)

3. DEA and star ranking

- DEA: Using predicted los and readmission rates as two inputs and volume as one output, physician's efficiency defined as: $\frac{v_1 * volume}{(w_1 * readmit_{adj} + w_2 * los_{adj})}$ is calculated by optimizing the weights with constraints such that with same weights applied to ach physician, efficiency cannot be greater than 1. (This is a relative measure.)
- Star ranking: This is nothing but a ranking based on 8 predetermined percentile. Physicians are ranked based on efficiency and per procedure.