This code takes the data from the NHSR and applies the algorithm in Kjaersgaard et al. (2016). The algorithm consists of nine steps. The input dataset is named Sysi.dta and has input variables: PNR (personal id) YNR (provider id) C_SPECIALE (provider speciality) 2 char string C_YDELSESNR (service provided) 4 char string V_HONAAR (year service was provided) 4 digit int V_HONUGE (week service was provided) int in [1,53] V_AFRAAR (year service was invoiced) 4 digit int V_AFRMDR (month service was invoiced) int in [1,12] C_TIDSKODE (time of service during week) 1 char string C_PATGRP (type of patient) 1 or 2 char string C_SIKGRP (health insurance group) 1 char string C_SIKKON (service in children indicator) 1 char string Furthermore, the code takes datasets with information on: (1) patient's date of death (Deathdates.dta) with variables PNR (personal id) ddate (date of death) created from C_STATUS and D_STATUS_HEN_START (2) patient's migrating between Denmark and other countries including Greenland (Migration.dta) with variables PNR (personal id) emi_from (date of emigration) emi_until (date of immigration) created from D_UDREJSE_DATO and D_INDREJSE_DATO or similar information in other variables (3) patient's moving between Danish municipalities (codes 101-861) (Moving.dta) with variables PNR (personal id) live_from (date patient moved to the municipality) live_until (date patient moved away) created from D_TILFLYT_DATO, D_FRAFLYTDATO, C_ANNKOR and C_KOM or similar information in other variables (4) practice closure identified as date of last service according to the NHSR (Practiceclosing.dta) with variables YNR (provider id) lastservice (date of last service) Variables with lowercase variable names need to be generated. Datasets are in long format with possible multiple records per person in datasets Migration.dta and Moving.dta. Please note that before December 31, 2006, Denmark was divided into 275 municipalities. The number was reduced to the current 98 on January 1, 2007. We used the codes of the former 275 municipalities.

--------------------------

Specify end of follow-up.
local enddate=mdy(12,31,2007)

tempfile Table2
#delimit ;
clear; input str4 C_YDELSESNR;
0101;0102;0103;0104;0105;0106;0107;0108;0109;0110;0120;0121;0122;0201;0202;0203;0411;0421;0431;0441;0451;0561;0491;0403;0401;0422;0423;0424;0425;0426;0427;0439;0463;0406;0427;0424;0429;6101;8110;8120;8130;8140;8142;8143;8144;8145;8146;8147;8148;8150;8160;8201;8202;8203;8204;8205;8206;8207;8208;8210;8211;8212;8213;8214;8215;8216;8217;8310;8317;8318;8319;8320;8326;8327;8328;8329;8330;8334;8335;8336;8701;8702;8703;8704;8705;8706;8707;8708;8901;8920;8921;8922;8923;8924;8925;8935;8936;8937;8938;
end;
### Step 1: Restrict services and recode time of services

* General practice services in Group 1 insured own patients
  * during regular work days and hours. Exclude services in
  * children reported with the personal identification number
  * of an adult.

```stata
use Sysi.dta, clear
keep if C_SPECIALE=="80" & C_TIDSKODE=="1" & C_SIKGRP=="1"
keep if C_PATGRP=="1" | C_PATGRP=="01"
drop if C_SIKKON=="B"
```

* Restrict services to basic fee services (Table 2).

```stata
merge m:1 C_YDELSESNR using `Table2', keep(match) nogen
```

* Recode time of services.

```stata
replace V_HONAAR=V_HONAAR-1 ///
if V_HONAAR==V_AFRAAR & inrange(V_HONUGE,51,53) & V_AFRAAR<=2
```

* Code date of treatment as Wednesday of week service
  * was provided.

```stata
gen wednesday_week1=3-dow(mdy(01,01,V_HONAAR))
gen treatdate=mdy(01,01,V_HONAAR)+wednesday_week1+7*(V_HONUGE-1)
replace treatdate=treatdate+7 ///
format treatdate %td
```

* Clean up.

```stata
keep  PNR V_HONAAR V_HONUGE treatdate YNR
order PNR V_HONAAR V_HONUGE treatdate YNR
label var V_HONAAR "Year of GP service"
label var V_HONUGE "Week of GP service"
label var treatdate "Wednesday of week of GP service"
label var YNR "Practice identification number"
```

### Step 2: Exclude services coded during

* emigration and after death

```stata
merge m:1 PNR using Deathdates.dta, keep(master match) nogen
```

* Exclude services coded after death.

```stata
drop if treatdate>=ddate
```

* Generate observation id.

```stata
gen long id=_n
```

* Merge data with information on emigration.

```stata
joinby PNR using Migration.dta, unmatched(master)
```

* Identify services coded during emigration.

```stata
gen ind=(emi_from<=treatdate & treatdate<emi_until)
bysort id: egen ind2=max(ind)
```

* Exclude services coded during emigration.

```stata
drop if ind2==1
```

* Clean up.

```stata
by id: keep if _n==1
drop id _merge emi* ind*
```

### Step 3: Exclude service weeks with multiple practices

*
* Identify service weeks with multiple practices.
bysort PNR V_HONAAR V_HONUGE (YNR): gen ind=(YNR[_n]!=YNR[1])
by PNR V_HONAAR V_HONUGE: egen ind2=max(ind)
* Exclude service weeks with multiple practices.
drop if ind2==1
* Clean up.
drop ind*

*************************************************** ***************
* Step 4: Code preliminary practice time intervals
*************************************************** ***************
* Code start date as the first contact with the practice.
bysort PNR (V_HONAAR V_HONUGE): gen GP_from=treatdate ///
   if _n==1 | (YNR[_n]!=YNR[_n-1] & _n>1)
* Clean up.
drop if GP_from==.
keep PNR YNR ddate GP_from
* Code end date as the minimum of day before contact at 
* another practice and end of follow-up.
by PNR: gen GP_until=min(GP_from[_n+1]-1,`enddate')
format GP_from GP_until %td

*************************************************** ***************
* Step 5: Recode practice time intervals 
* taking into account emigration and death
*************************************************** ***************
* Generate observation id.
gen long id=_n
* Merge data with information on emigration.
joinby PNR using Migration.dta, unmatched(master)
* Identify practice time intervals with emigration.
gen temp=emi_from if GP_from<=emi_from & emi_from<=GP_until 
bysort id: egen temp2=min(temp)
* Recode end date as the minimum of day before contact at 
* another practice, end of follow-up, date of emigration or death.
replace GP_until=min(GP_until,temp2,ddate)
* Clean up.
by id: keep if _n==1
drop id _merge emi* temp*

*************************************************** ***************
* Step 6: Recode practice time intervals 
* taking into account practice closure
*************************************************** ***************
* Merge data with information on practice closure.
merge m:1 YNR using Practiceclosing.dta, keep(match master) nogen
* Identify practice time intervals with practice closure.
gen ind=(GP_from<=lastservice & lastservice<=GP_until)
bysort PNR (GP_from):
    gen ind2=(GP_from[_n-1]<=lastservice[_n-1] ///
        & lastservice[_n-1]<=GP_until[_n-1] ///
        & GP_from[_n]==GP_until[_n-1]+1 & _n>1)
* Recode end date to the date of closure and recode start date 
* for the subsequent interval to the next day.
replace GP_until=lastservice if ind==1
replace GP_from=lastservice[_n-1]+1 if ind2==1
* Clean up.
drop last* ind*

*************************************************** ***************
* Step 7: Recode practice time intervals 
* taking into account patients moving
*************************************************** ***************
* Generate observation id.  
gen long id=_n  
* Merge data with information on patients moving.  
joinby PNR using Moving.dta, unmatched(master)  
* Identify GP intervals with patient moving.  
gen temp=live_from if GP_from<=live_from & live_from<=GP_until  
bysort id: egen temp2=max(temp)  
gen temp3=temp2[_n-1]+1 if GP_from[_n]==GP_until[_n-1]+1 & _n>1  
by id: replace temp3=temp3[_n-1] if _n>1  
* Recode end date to the date of moving and recode start date  
* for the subsequent interval to the next day.  
replace GP_until=temp2 if temp2<.  
replace GP_from=temp3 if temp3<.  
* Clean up.  
by id: keep if _n==1  
drop id _merge live* temp*  
***************************************************  
* Step 8: Drop small (<31 days) practice time intervals  
***************************************************  
* Identify last date for several consecutive practice intervals.  
bysort PNR (GP_from): gen temp_until=GP_until  
if _n==N | (GP_until[_n]<GP_from[_n+1]-1 & _n<=N)  
gsort +PNR -GP_until  
by PNR: replace temp_until=temp_until[_n-1]  
if temp_until==. & _n>1  
sort PNR GP_from  
* Drop practice time intervals < 31 days.  
drop if GP_until-GP_from<31  
by PNR: replace GP_until=GP_from[_n+1]-1  
if GP_from[_n]==GP_from[_n+1] & _n<_N  
by PNR: replace GP_until=temp_until[_n]  
if _n==N | (temp_until[_n]!=temp_until[_n+1] & _n<_N)  
drop temp_until  
***************************************************  
* Step 9: Recode practice time intervals to monthly intervals  
***************************************************  
* Recode start date to the first day of month.  
gen GP_from_month=mdy(month(GP_from),01,year(GP_from))  
* Unless the interval ends on the date of death or emigration  
* or on the last day in month, recode end date to the last day  
* of the preceding month.  
gen GP_until_month=ddate if ddate==GP_until  
bysort PNR (GP_from): replace GP_until_month=GP_until  
if GP_from[_n]==GP_from[_n+1] & _n<_N  
replace GP_until_month=GP_until  
if month(GP_until)!=month(GP_until+1)  
replace GP_until_month=mdy(month(GP_until),01,year(GP_until))-1  
if GP_until_month==.  
format GP_from_month GP_until_month %td  
drop ddate  
***************************************************  
* Save patient list (Patientlist.dta). The saved dataset will  
* contain the following variables:  
* PNR  
* YNR  
* GP_from (starting date before Step 9)  
* GP_from_month (starting date after Step 9)  
* GP_until (end date before Step 9)  
* GP_until_month (end date after Step 9)  
***************************************************  
save Patientlist.dta