Children's Bureau (CB)

Child and Family Services Review (CFSR) Round 3

Statewide Data Indicators: PDF Version of Syntax

Revised Syntax Pending Final Verification

September 2018

Table of Contents

CFSR 3: CREATE CENSUS CHILD POPULATIONS BY STATE 3

CFSR 3: CREATE AFCARS SOURCE DATA 7

CFSR 3: CREATE NCANDS SOURCE DATA 26

CFSR 3: CREATE AFCARS DATA QUALITY (DQ) FILES 30

CFSR 3: CREATE NCANDS DATA QUALITY (DQ) FILES 34

CFSR 3: OBSERVED PERFORMANCE FOR MALTREATMENT IN FOSTER CARE 41

CFSR 3: OBSERVED PERFORMANCE FOR RECURRENCE OF MALTREATMENT 67

CFSR 3: OBSERVED PERFORMANCE - PERMANENCY IN 12 MONTHS FOR CHILDREN ENTERING FOSTER CARE and RE-ENTRY TO FOSTER CARE IN 12 MONTHS 78

CFSR 3: OBSERVED PERFORMANCE - PERMANENCY IN 12 MONTHS FOR CHILDREN IN CARE ON FIRST DAY 12-23 MONTHS, and PERMANENCY IN 12 MONTHS FOR CHILDREN IN CARE ON FIRST DAY 24 MONTHS OR MORE 98

CFSR 3: OBSERVED PERFORMANCE - PLACEMENT STABILITY 115

CFSR 3: RISK-STANDARDIZED PERFORMANCE - MALTREATMENT IN FOSTER CARE 131

CFSR 3: RISK-STANDARDIZED PERFORMANCE - RECURRENCE OF MALTREATMENT 142

CFSR 3: RISK-STANDARDIZED PERFORMANCE - PERMANENCY IN 12 MONTHS FOR CHILDREN ENTERING FOSTER CARE 152

CFSR 3: RISK-STANDARDIZED PERFORMANCE - PERMANENCY IN 12 MONTHS FOR CHILDREN IN CARE ON FIRST DAY 12-23 MONTHS 162

CFSR 3: RISK-STANDARDIZED PERFORMANCE - PERMANENCY IN 12 MONTHS FOR CHILDREN IN CARE ON FIRST DAY 24 MONTHS OR MORE 172

CFSR 3: RISK-STANDARDIZED PERFORMANCE - PLACEMENT STABILITY 182

CFSR 3: RISK-STANDARDIZED PERFORMANCE - RE-ENTRY TO FOSTER CARE IN 12 MONTHS 193

CFSR 3: CREATE CENSUS CHILD POPULATIONS BY STATE

```
* Encoding: UTF-8.

************************************************
CFSR 3: CHILD POPULATIONS BY STATE
************************************************

* This syntax is being provided for those interested in learning how CB
obtains child population data, and for
* building states' capacity to produce these data moving forward. Child
population estimates are used to
* calculate states' entry rates. Entry rate is used as a risk adjustment
variable for two indicators: Permanency
* in 12 months for children entering foster care and Re-entry to foster
care in 12 months. For details on how entry
* rates are calculated, see the Federal Register notice and the relevant
syntax used to calculate observed
* performance for the two indicators.

* Note: A file containing child population estimates for years 2011,
2012, 2013, 2014, 2015, and 2016 is already available in
* the CFSR3IndicatorSyntax.zip file. When that file is unzipped, it will
be saved at
* the Fixed files directory. The file includes child population estimates
for all 50 states, the
* District of Columbia and Puerto Rico. These historical estimates are
based on Census
* estimates with a release date of June 2017, using the data available
form the Census website.
* CB selected only records where AGE was less than 18, SEX was equal to 0
(0 = males and females), and ORIGIN was equal to 0 (0 =
* all origins), and then summed the population estimates for each state,
for each year available in the file.

************************************************

* Population estimates for years 2017 and beyond can be processed using
the syntax below, which will also
* append each new year of data to the existing Child populations.sav
file. The syntax in this example uses the
* 2016 census year as an example, even though at the time this syntax is
written that file is not yet available.
*The set of macros at the beginning may need to be changed to match the
year being
* processed.

* 1.  Go to http://www.census.gov/popest/data/index.html
* 2.  For States (Level of Geography), Population by age, sex, race, and
Hispanic origin (Level of Detail), click the
*     link under Most Current Data (e.g., V2016)
* 3.  Under Downloadable Datasets, click the link for Population
estimates by age (18+), select the CSV link and save the data to
*     your computer (e.g., C:\CFSR3\) as a CSV file. This file (e.g.,
SCPRC-EST2016-18+POP-RES.csv) contains
*     population data for all 50 states, D.C., and Puerto Rico.
```

```
*In all four macros, change the year to reflect the Census year shown in
the file you downloaded.
define YY () 16
!enddefine.
define YYYY () 2016
!enddefine.
define POPESTIMATE () POPESTIMATE2016
!enddefine.
define POPEST18PLUS () POPEST18PLUS2016
!enddefine.


* Root directory.
file handle Root_FH /name="C:\CFSR 3\Analysis - DP 2017 July".

* Directory for the Census data file.
file handle Census_FH /name="Root_FH\0 - Source Data\Census".

* Folder where child population files are located (fixed files are
generally useful static files that are used in various different sets of
code).
file handle Fixed_FH /name="Root_FH\1 - Fixed files".

GET DATA  /TYPE=TXT
  /FILE= 'Census_FH\SCPRC-EST2016-18+POP-RES.csv'
  /DELCASE=LINE
  /DELIMITERS=","
  /ARRANGEMENT=DELIMITED
  /FIRSTCASE=2
  /IMPORTCASE=ALL
  /VARIABLES=
  SUMLEV F3.0
  REGION A1
  DIVISION A1
  STATE F2.0
  NAME A24
  POPESTIMATE2016 F9.0
  POPEST18PLUS2016 F9.0
  PCNT_POPEST18PLUS F4.1.
CACHE.
EXECUTE.
DATASET NAME CurrentYear WINDOW=FRONT.

compute ReleaseDate = date.dmy(01,06,YY+1).
formats ReleaseDate (adate10).
execute.

* 6.  Subtract POPEST18PLUS from POPESTIMATE. This yields the child
populations (17 and under) for all 50 states, D.C., and Puerto Rico.

compute ChildPopulation = POPESTIMATE - POPEST18PLUS.
```

```
execute.

* 7.  Delete the record where STATE = 0, which holds the total population
of the United States.

select if STATE <> 0.
execute.

* 8. Create variable holding the census year the population estimate is
based on (e.g., July 1st, YYYY).
*     Census estimates are always based on July 1st of the given year.

compute PopYear = date.dmy(01, 07, YYYY).
formats PopYear (adate10).
execute.

* 9.  Append the new year's estimates for this release to the existing
Child populations.sav file.

get file 'Fixed_FH\Child populations.sav'.
dataset name PastYears window=front.
dataset activate PastYears.
add files /FILE=*
  /FILE='CurrentYear'
  /RENAME (DIVISION NAME PCNT_POPEST18PLUS POPEST18PLUS POPESTIMATE
REGION SUMLEV=d0 d1 d2
    d3 d4 d5 d6)
  /DROP=d0 d1 d2 d3 d4 d5 d6.
execute.
dataset close CurrentYear.
sort cases by STATE PopYear.
save outfile 'Fixed_FH\Child populations.sav'.
```

CFSR 3: CREATE AFCARS SOURCE DATA

```
* Encoding: UTF-8.
set printback=on.


************************************************
* MERGE 6-MONTH SUBMISSIONS
************************************************

* Calculating performance on CFSR 3 indicators requires multiple 6-month
submissions. The number of
* submissions needed depends on the indicator. The date of those
submissions (i.e., the reporting
* period) depends on the the 12-month cohort for which performance will
be measured. For example, to
* measure performance for Permanency in 12 months for children entering
foster care (and the companion
* Re-entry indicator) requires six, 6-month submissions (i.e., three
years of data). So, to measure Permanency
* for children who entered during 11B12A requires these 6-month
submissions: 11B, 12A, 12B, 13A, 13B, and
* 14A.

* This step will merge together all 6-month submissions stored in the
'AFCARS Submissions' folder. In determining
* what 6-month files to include, CB recommends starting with 09B and
ending with the most recent submission
* available (i.e., from the latest reporting period). In other words, CB
recommends building and maintaining
* a historical dataset starting with 09B (although there is no problem
with adding earlier periods).

* Merge together all files in the submissions folder.
************************************************

* Root directory. ****PLEASE CHANGE THIS BASED ON WHERE YOU UNZIP AND
SAVE THE CFSR 3 FOLDER.****.
file handle Root_FH /name="C:\CFSR 3".

***DO NOT CHANGE FILE HANDLES BELOW****.
*Directory to save combined AFCARS file.
file handle AFCARS_Comb_FH/name='Root_FH\0 - Source Data'.

*Directory for all AFCARS Submissions.
file handle AFCARS_Source_FH/name='AFCARS_Comb_FH\Submissions AFCARS'.

* Folder where fixed files are located (fixed files are generally useful
static files that are used in various different sets of code).
file handle Fixed_FH /name="Root_FH\1 - Fixed files".

* Merge files. This syntax requires that SPSS Python Essentials be
installed. It is installed by default
* beginning with SPSS Version 22.

begin program.
```

```
import spss, spssaux, os, glob

fh = spssaux.FileHandles()

rdir = fh.resolve('AFCARS_Source_FH')  # Specifies folder containing .sav
files.
fils = sorted([fil for fil in os.listdir(rdir) if fil.endswith('.sav')])
spss.Submit('get file "%s/%s".'%(rdir,fils.pop(0)))
for rep in range(len(fils)/49 + 1):
    spss.Submit('add files
file=*/%s.'%'/'.join(['file="%s"'%os.path.join(rdir,fil) for fil in
fils[49*rep:49*rep + min(49,len(fils)-49*rep)]]))
spss.Submit('exe.')
end program.

dataset name SourceFile.
Cache.
EXECUTE.
*************************************************
* FILE PREP
*************************************************

* Create a unique, numeric ID (Child ID) across the entire file.
*************************************************

* Create a unique ID (recnumbr2) across the entire file (state FIPS +
recnumber).
sort cases by state recnumbr.
string statestring (A2).
compute statestring = STRING(state, F2.0).

string recnumbr2 (A50).
compute recnumbr2=concat(statestring,recnumbr).


* Create a unique, numeric version of recnumbr2, called ChildID. ChildID
will be used later for sorting, breaking,
* and other functions. It is more efficient and reliable than using the
recnumbr2, which is a string and, for some
* states, contains highly specialized characters.
autorecode VARIABLES=recnumbr2
   /INTO ChildID.
value labels ChildID.
delete variables statestring recnumbr2.

* Add state names and abbreviations.
*************************************************

* Add state full names and abbreviations by merging these in from
states.sav data file. These are useful for
* reporting purposes, custom tables, etc.
get file 'Fixed_FH\states.sav'.
dataset name states window=front.
sort cases by state.
```

```
save outfile='Fixed_FH\states.sav'.
dataset activate SourceFile.
dataset close states.
match files /FILE=*
  /TABLE='Fixed_FH\states.sav'
  /BY state.
execute.



* Create date fields.
*************************************************.

* Start and end dates for each 6-month reporting period in the merged
file.
* ... A files (Oct 1 - Mar 31).
if repdatmo = 3 DtReportBeg = date.mdy(10,1,repdatyr-1).
if repdatmo = 3 DtReportEnd= date.mdy(3,31,repdatyr).
* ... B files (Apr 1 - Sep 30).
if repdatmo = 9 DtReportBeg = date.mdy(4,1,repdatyr).
if repdatmo = 9 DtReportEnd = date.mdy(9,30,repdatyr).

* Date of the last 6-month reporting period in the merged file.
aggregate
  /OUTFILE=* MODE=ADDVARIABLES
  /BREAK=
  /DtReportEndFinal=MAX(DtReportEnd).

* Other key dates. Some of these are not used in the CFSR syntax, and are
created for convenience in case
* users are interested in making use of them).
compute DtReview=date.mdy(pedrevmo, pedrevda,pedrevyr).
compute DtBirth=date.mdy(dobmo,dobda,dobyr).
compute DtFirRem=date.mdy(rem1mo,rem1da,rem1yr).
compute DtPriorDisch =date.mdy(dlstfcmo,dlstfcda,dlstfcyr).
compute DtLatRem=date.mdy(latremmo,latremda,latremyr).
compute DtLatRemTrans=date.mdy(remtrnmo, remtrnda, remtrnyr).
compute DtCurSet=date.mdy(cursetmo,cursetda,cursetyr).
compute DtTPRMom=date.mdy(prtmommo, prtmomda, prtmomyr).
compute DtTPRDad=date.mdy(prtdadmo, prtdadda, prtdadyr).
compute DtDisch=date.mdy(dodfcmo,dodfcda,dodfcyr).
compute DtDischTrans=date.mdy(dodtrnmo,dodtrnda,dodtrnyr).
execute.

* Formatting.
variable labels
DtReportBeg 'Start date of the 6-month reporting period (10/1/YYYY or
4/1/YYYY)'
DtReportEnd 'End date of the 6-month reporting period (3/31/YYYY or
9/30/YYYY)'
DtReportEndFinal 'Date of the last 6-month reporting period in the file'
DtReview 'Date of most recent periodic review'
DtBirth 'Date of birth'
DtFirRem 'Date of first removal from home'
```

```
DtPriorDisch 'Date of discharge from last foster care episode'
DtLatRem 'Date of latest removal from home'
DtLatRemTrans 'Date (transaction) of latest removal from home'
DtCurSet 'Date of placement in current foster care setting'
DtTPRMom 'Date of mother parental rights termination'
DtTPRDad 'Date of father parental rights termination'
DtDisch 'Date of discharge from foster care'
DtDischTrans 'Date (transaction) of discharge from foster care'.
formats DtReportBeg to DtDischTrans (adate10).

* Delete original date fields.
delete variables
repdatyr repdatmo
pedrevyr pedrevmo pedrevda
dobyr dobmo dobda
rem1yr rem1mo rem1da
dlstfcyr dlstfcmo dlstfcda
latremyr latremmo latremda
remtrnyr remtrnmo remtrnda
cursetyr cursetmo cursetda
prtmomyr prtmommo prtmomda
prtdadyr prtdadmo prtdadda
dodfcyr dodfcmo dodfcda
dodtrnyr dodtrnmo dodtrnda.

* Delete duplicate records.
************************************************

* Identify children reported two or more times during the same 6-month
reporting period. These occur rarely, as
* states are advised to report only the most recent episode per 6-month
reporting period.
SORT CASES BY ChildID(A) DtReportBeg(A) DtLatRem(A) DtDisch(A).
MATCH FILES
  /FILE=*
  /BY ChildID DtReportBeg
  /FIRST=PrimaryFirst
  /LAST=PrimaryLast.
DO IF (PrimaryFirst).
COMPUTE  MatchSequence=1-PrimaryLast.
ELSE.
COMPUTE  MatchSequence=MatchSequence+1.
END IF.
LEAVE  MatchSequence.
FORMATS  MatchSequence (f7).
COMPUTE  InDupGrp=MatchSequence>0.
SORT CASES InDupGrp(D).
MATCH FILES
  /FILE=*
  /DROP=PrimaryFirst MatchSequence.
VARIABLE LABELS  PrimaryLast 'Indicator of each last matching case as
Primary'.
VALUE LABELS  PrimaryLast 0 'Duplicate Case' 1 'Primary Case'.
VARIABLE LEVEL  PrimaryLast (ORDINAL).
```

```
EXECUTE.

* Document states that reported the same children two or more times in a
report period, and the number of
* duplicates reported.
temporary.
select if InDupGrp = 1 and PrimaryLast = 0.
EXECUTE.

* Discard duplicates. The sort order means we try to keep the record with
the latest removal date (if it differs)
* and, among those, the record with a discharge date (if present).
select if PrimaryLast = 1.
execute.

delete variables PrimaryLast InDupGrp.

*Add rptyr and rptmo variables needed for later merge with NCANDS data.

compute rptyr = xdate.year(dtreportbeg).
compute rptmo = xdate.month(dtreportbeg).
execute.

if (rptmo eq 10) FY eq rptyr + 1.
if (rptmo eq 4) FY eq rptyr.
execute.

string afcarsid (a36).
compute afcarsid= recnumbr.
execute.

delete variables rptyr rptmo.


* Delete records with no afcarsid.
************************************************

SELECT IF AFCARSID <> ''.
EXECUTE.

************************************************
* PREPARATION FOR DATA QUALITY - AFCARS CROSS-FILE CHECKS
************************************************

* Cross-file checks look across every two consecutive 6-month reporting
periods in the merged file to determine
* if a child was reported in the next 6-month report period. Two checks
are then carried out:

* 1. Dropped records
* 2. AFCARS IDs don't match from one period to the next

* These checks are not performed on records that occur in the last
reporting period in the merged file, because
```

```
* there is no subsequent 6-month reporting period to look for.

* Preparation - Calculate TimeBetweenReports.
************************************************

* First, calculate the time in months between the child's record and any
subsequent report that may exist for
* the child. The two DQ checks will use this information.
sort cases BY ChildID(A) DtReportBeg(A).

* For each record, make a copy of DtReportBeg, which stores the start
date of the 6-month reporting period.
compute DtReportBeg1 = DtReportBeg.
execute.
formats DtReportBeg1 (adate10).

* Create a variable called Next6MoReport. If the child was reported in a
subsequent 6-month period (i.e.,
* another DtReportBeg for this child is found), copy the start date of
that subsequent 6-month submission
* into Next6MoReport. If the child was not reported in a subsequent
submission, leave Next6MoReport blank.
* Note: For a new child, DtReportBeg1 is set to missing; otherwise, the
start date will will get assigned to the
* previous child.
PRESERVE.
set workspace 200000.
if (ChildID<>lag(ChildID)) DtReportBeg1 = $sysmis.
create Next6MoReport=Lead(DtReportBeg1,1).
execute.
formats Next6MoReport (adate10).
RESTORE.


* Calculate the number of months between the start date of the current 6-
month reporting period and the start
* date of the subsequent 6-month report period that was found for this
child. If the reporting period is the last
* reporting period in the merged file, then there is no subsequent report
to look for, so code that as 8888 (Not
* Applicable).

do if DtReportEnd eq DtReportEndFinal.
compute TimeBetweenReports = 8888.
else.
compute TimeBetweenReports =
datediff(Next6MoReport,DtReportBeg,"months").
end if.


MISSING VALUES TimeBetweenReports (7777, 8888, 9999).

execute.
```

```
* Possible results for TimeBetweenReports:
* ... blank = a subsequent report was not found for this child
* ... 6 = a subsequent report for this child was found, and it occurred
in the next 6-month reporting period
* ... 12 or more = a subsequent report for this child was found, but it
occurred in a later reporting period (i.e., not
*     the immediate next one).
* ... 8888 = not applicable (report is in the last reporting period in
the merged file).


*Create Age variable, using age at entry if child entered
* Age at entry (categorical), specified as 0-3 mos, 4-11 mos, 1-5 yrs, 6-
10 yrs, 11-16 yrs, 17, 18-21 yrs.
* Useful for reporting and descriptive statistics.
*******************************

* Calculate age at entry in months.
compute AgeNmos = datediff(DtLatRem,DtBirth,"months").
* Record age in months to desired categories.
recode AgeNmos (216 thru 263=6) (204 thru 216=5) (132 thru 204=4) (72
thru 132=3) (12 thru 72=2) (4 thru 12=1) (0 thru 4=0) (else=sysmis) into
AgeNmosyrsCat.
execute.

* Age at entry (interval), specified as 0-3 mos, 4-11 mos, 1 yr ... 21
yrs.
* Used later in risk adjustment, where each age value is converted to a
yes/no (1/0) dummy variable.
*******************************

* Calculate age at entry in years.
compute AgeNyrs = datediff(DtLatRem,DtBirth,"years").
* Children < 1 currently have a value of 0 for AgeNyrs. Need to split
these into 0-3 mos (0) mos and 4-11 mos (1).
* To make room for the 0 and 1 for < 1 children, make a copy of AgeNyrs
then shift all subsequent ages, so age1 = 2, age2 = 3, etc.
compute AgeNmosyrs = AgeNyrs.
* Recode 1 - 21 to 2 - 22.
recode AgeNyrs (1=2) (2=3) (3=4) (4=5) (5=6) (6=7) (7=8) (8=9) (9=10)
(10=11) (11=12) (12=13) (13=14) (14=15) (15=16) (16=17) (17=18) (18=19)
(19=20) (20=21) (21=22) INTO AgeNmosyrs.
* If child is 0-3, make AgeNmosyrs = 0.
* If child is 4-11, make AgeNmosyrs = 1.
if AgeNmosyrsCat = 0 AgeNmosyrs = 0.
if AgeNmosyrsCat = 1 AgeNmosyrs = 1.
execute.

* Calculate age on first day of report period in years.

compute AgeRPTBegyrs = datediff(DtReportBeg,DtBirth,"years").
execute.

compute ExAge18=0.
```

```
if AgeNyrs ge 18 or AgeRPTBegyrs ge 18 ExAge18=1.
exe.

delete variables AgeRPTBegyrs.

* Youth who turn 18 during the 12-month period will not have time in care
beyond their
* 18th birthday or moves after their 18th birthday counted. To handle
this, for children who
* turn 18 during the period we replace their discharge date with the date
they turned 18.
* Therefore, the LOS for children who turn 18 during the period will be
calculated from date of latest
* removal to date of 18th birthday.
************************************************

* Calculate child's 18th birthday.
compute Bday18=DATESUM(DtBirth, 18, "years", 'closest').
variable labels  Bday18 "18th birthday".
variable level Bday18(SCALE).
formats Bday18(ADATE10).
variable width Bday18(10).
execute.

* If 18th birthday occurred during the 12-month period, and child had
discharged by the time he turned 18,
* replace date of discharge with date of 18th birthday.
compute Adjust18=0.
EXECUTE.
do if (Bday18 gt DtReportBeg and Bday18 le DtReportEnd) and
(missing(DtDisch) or DtDisch gt Bday18).
compute DtDischAdjusted=Bday18.
compute Adjust18=1.
end if.
execute.
variable labels DtDischAdjusted 'For children who turned 18 during the
period and had not discharged by that time, the date of their 18th
birthday is their effective date of discharge'.

* Copy date of discharge for remaining children to DtDischAdjusted.
do if sysmis(DtDischAdjusted).
compute DtDischAdjusted=DtDisch.
end if.
execute.
formats DtDischAdjusted (adate10).

* Dropped records.
************************************************

* It is a dropped record if the 6-month record is missing a DtDisch and a
subsequent report was not found for
* this child (TimeBetweenReports = BLANK) or a subsequent report was
found for this child but it occurred in
```

```
* a reporting period other than the immediate next one
(TimeBetweenReports >= 12 months).
*Dropped case check should be blank  if child was 18 on first day of the
period or at entry or turned 18 during the period.
do if (ExAge18=0)  and (Adjust18=0).
    compute DQ_Dropped=0.
    do if sysmis(DtDisch).
        if sysmis(TimeBetweenReports) OR (TimeBetweenReports ge 12 AND
TimeBetweenReports lt 8888) DQ_Dropped = 1.
    end if.
end if.
execute.


* AFCARS IDs don't match from one period to next.
**************************************************

* It is a non-match if the child was not reported in the next 6-month
period (TimeBetweenReports = BLANK) or a
* a subsequent report was found for this child but it occurred in a
reporting period other than the immediate next
* one (TimeBetweenReports >= 12 months). These records are not in error
and not a problem unless the
* percentage of non-matches for the entire state exceeds the DQ limit.
do if (ExAge18=0)  and (Adjust18=0).
    compute DQ_IDNoMatchNext6Mo=0.
    if sysmis(TimeBetweenReports) OR (TimeBetweenReports ge 12 AND
TimeBetweenReports lt 8888) DQ_IDNoMatchNext6Mo = 1.
end if.
execute.

delete variables DtReportBeg1 Next6MoReport TimeBetweenReports.


**************************************************
* PREPARATION FOR DATA QUALITY - AFCARS WITHIN-FILE CHECKS
**************************************************

* These checks are run on each 6-month record in the combined file,
within each 6-month report period.
* #variables are scratch variables - they are created temporarily and
then automatically removed.

* Missing date of birth.
* Missing date of latest removal.
* Missing number of placement settings.

compute DQ_missDOB=0.
if sysmis(DtBirth) DQ_missDOB = 1.

*Limit the following DQ checks to children under 18.

do if ExAge18=0.
compute DQ_missDtLatRem=0.
```

```
if sysmis(DtLatRem) DQ_missDtLatRem = 1.
compute DQ_missNumPlep=0.
if sysmis(numplep) DQ_missNumPlep = 1.
* Date of birth is after the date of latest removal (produces a negative
age of entry).
if not(missing(DtBirth)) and not(missing(DtLatRem)) DQ_DOBgtDtLatRem = 0.
if DtBirth > DtLatRem DQ_DOBgtDtLatRem = 1.
* Date of birth is after date of discharge (produces a negative age at
exit).
if not(missing(DtBirth)) and not(missing(DtDisch)) DQ_DOBgtDtDisch = 0.
if DtBirth > DtDisch DQ_DOBgtDtDisch = 1.
* Date of latest removal is equal to the date of discharge (produces a
LOS of 0 to 24 hrs).
if not(missing(DtDisch)) and not(missing(DtLatRem))
DQ_DtDischeqDtLatRem=0.
if DtDisch = DtLatRem DQ_DtDischeqDtLatRem = 1.
* Date of discharge is before the date of latest removal (produces a
negative LOS).
if not(missing(DtDisch)) and not(missing(DtLatRem))
DQ_DtDischltDtLatRem=0.
if DtDisch < DtLatRem DQ_DtDischltDtLatRem = 1.
* Date of discharge exists but the reason for discharge is missing.
if not(missing(DtDisch)) and not(missing(disreasn))  DQ_missDisreasn=0.
if not(missing(DtDisch)) and missing(disreasn) DQ_missDisreasn = 1.
* Child is on first removal (total # of removals from home to date = 1).
* Not a problem unless the % of children on their first removal exceeds
the DQ limit.
if not(missing(totalrem)) DQ_totalrem1=0.
if totalrem = 1 DQ_totalrem1 = 1.
end if.
execute.

*Include all children include those over 18 for these.
* Time between date of birth and date of latest removal is > 21 yrs
(produces age at entry > 21 yrs).
if not(missing(DtBirth)) and not(missing(DtLatRem))
DQ_gt21DOBtoDtLatRem=0.
compute #AgeN = datediff(DtLatRem,DtBirth,"years").
if #AgeN > 21 DQ_gt21DOBtoDtLatRem = 1.
* Time between date of birth and date of discharge is > 21 yrs (produces
age at exit > 21 yrs).
if not(missing(DtBirth)) and not(missing(DtDisch)) DQ_gt21DOBtoDtDisch =
0.
compute #AgeX = datediff(DtDisch,DtBirth,"years").
if #AgeX > 21 DQ_gt21DOBtoDtDisch = 1.
* Time between date of latest removal and date of discharge is > 21 yrs
(produces a LOS > 21 yrs).
* MHMH changed datediff  from days to years.
if not(missing(DtDisch)) and not(missing(DtLatRem))
DQ_gt21DtDischtoDtLatRem = 0.
compute #LOSN = datediff(DtDisch, DtLatRem, "years").
if #LOSN > 21 DQ_gt21DtDischtoDtLatRem = 1.
EXECUTE.
*added flag for dtpriordisch after dtlatrem.
```

```
compute DQ_dtpriordafterdlm = 0.
if (dtpriordisch gt dtlatrem) DQ_dtpriordafterdlm = 1.
execute.

* Formatting
*************************************************
* As noted earlier, records with a 1 for DQ_IDNoMatchNext6Mo are not in
error and not a problem
* unless the percentage of non-matches for the entire state exceeds the
DQ limit.
value labels DQ_Dropped to DQ_totalrem1
0 'Okay'
1 'Potential problem'.
formats DQ_Dropped to DQ_totalrem1 (F2.0).

variable labels
DQ_Dropped 'Record is missing a date of discharge, suggesting child is
still in care, but a record in the next 6-month period does not exist'
DQ_IDNoMatchNext6Mo 'No match for a given recnumbr in the next 6-month
period. Not a problem unless the percentage of non-matches for a given
state exceeds the DQ limit'
DQ_missDOB 'Date of birth is missing'
DQ_missDtLatRem 'Date of latest removal from home is missing'
DQ_missNumPlep 'Number of placements in current foster care episode is
missing'
DQ_DOBgtDtLatRem 'Date of birth is after the date of latest removal from
home (produces a negative age of entry)'
DQ_DOBgtDtDisch 'Date of birth is after the date of discharge from most
recent foster care episode (produces a negative age at exit)'
DQ_gt21DOBtoDtLatRem 'Time between date of birth and date of latest
removal from home is > 21 yrs (produces age at entry > 21 yrs)'
DQ_gt21DOBtoDtDisch 'Time between date of birth and date of discharge
from foster care is > 21 yrs (produces age at exit > 21 yrs)'
DQ_gt21DtDischtoDtLatRem 'Time between date of latest removal from home
and date of discharge from most recent foster care episode is > 21 yrs
(produces a LOS > 21 yrs)'
DQ_DtDischeqDtLatRem 'Date of latest removal from home is the same day as
the date of discharge from most recent foster care episode (produces a
LOS of 0 days)'
DQ_DtDischltDtLatRem 'Date of discharge from most recent foster care
episode is before the date of latest removal from home (produces a
negative LOS)'
DQ_missDisreasn 'Date of discharge from most recent foster care episode
exists but the reason for discharge is missing'
DQ_totalrem1 'Child is on first removal (total # of removals from home to
date = 1). Not a problem unless the percentage of children on their first
removal for a given state exceeds the DQ limit'
DQ_dtpriordafterdlm 'Date of prior discharge is after date of latest
removal in same record'.

*************************************************
* CALCULATE DERIVED VARIABLES
*************************************************
```

```
* These derived variables are calculated for each record in the file.
Some will be used in the indicator-specific
* syntax; others may be useful for other reporting and analyses.

* Entry and exit years
************************************************
* FFY child entered.

compute EntryYr=xdate.year(DtLatRem).
if range(xdate.month(DtLatRem),10,12) EntryYr=EntryYr+1.
* If date of latest removal is missing, recode to 9999.
if DQ_missDtLatRem = 1 EntryYr = 9999.
execute.

* FFY child exited.
compute ExitYr=xdate.year(DtDisch).
if range(xdate.month(DtDisch),10,12) ExitYr=ExitYr+1.
* If date of discharge is missing, assume child is still in care.
if sysmis(DtDisch) ExitYr = 7777.
execute.

MISSING VALUES EntryYr ExitYr (7777, 8888, 9999).

variable labels
EntryYr 'FFY child entered'
ExitYr 'FFY child exited care (or 7777 if child is still in care)'.
value labels
EntryYr 9999 'Cannot calculate (DtLatRem missing)'
/ ExitYr 7777 'Child still in care (DtDisch missing)'.
formats EntryYr ExitYr (F4.0).

* Handle problem values.
*******************************

* If age at entry is negative or > 21 yrs, recode to 8888.
do if DQ_DOBgtDtLatRem = 1 or DQ_gt21DOBtoDtLatRem = 1.
recode AgeNmos AgeNyrs AgeNmosyrsCat AgeNmosyrs (else = 8888).
end if.
* If age at entry can't be calculated, recode to 9999 (missing DOB or
DtLatRem).
do if  DQ_missDOB = 1 or DQ_missDtLatRem = 1.
recode AgeNmos AgeNyrs AgeNmosyrsCat AgeNmosyrs (else = 9999).
end if.

* 9-28-2016 - MH adding missing value assignment to 7777,8888,9999.
MISSING VALUES AgeNmos AgeNyrs AgeNmosyrsCat AgeNmosyrs (7777, 8888,
9999).
execute.

* Age at exit (categorical), specified as 0-3, 4-11, 1-5, 6-10, 11-16,
17, 18-21.
* Useful for reporting and descriptive statistics.
*******************************
```

```
* Calculate age at exit in months (with 777 for still in care).
compute AgeXmos = datediff(DtDisch,DtBirth,"months").
if sysmis(DtDisch) AgeXmos = 7777.
* Recode age in months to desired categoies.
recode AgeXmos (7777=7777) (216 thru 263=6) (204 thru 216=5) (132 thru
204=4) (72 thru 132=3) (12 thru 72=2) (4 thru 12=1) (0 thru 4=0)
(else=sysmis) into AgeXmosyrsCat.
execute.

* 9-28-2016 - MH adding missing value assignment to 7777,8888,9999.
MISSING VALUES AgeXmos (7777, 8888, 9999).

* Age at exit (interval), specified as 0-3 mos, 4-11 mos, 1 yr ... 21
yrs.
* Used later in risk adjustment, where each age value is converted to a
yes/no (1/0) dummy variable.
*********************************

* Calculate age at exit in years (with 777 for still in care)..
compute AgeXyrs = datediff(DtDisch,DtBirth,"years").
if sysmis(DtDisch) AgeXyrs = 7777.
execute.
* Children < 1 currently have a value of 0. Need to split these into 0-3
mos (0) mos and 4-11 mos (1).
* To make room for the 0 and 1 for < 1 children, make a copy of AgXNyrs
then shift all subsequent ages, so age1 = 2, age2 = 3, etc.
compute AgeXmosyrs = AgeXyrs.
* Recode 1 - 21 to 2 - 22.
recode AgeXyrs (1=2) (2=3) (3=4) (4=5) (5=6) (6=7) (7=8) (8=9) (9=10)
(10=11) (11=12) (12=13) (13=14) (14=15) (15=16) (16=17) (17=18) (18=19)
(19=20) (20=21) (21=22) INTO AgeXmosyrs.
* If child is 0-3, make AgeXmosyrs = 0.
* If child is 4-11, make AgeXmosyrs = 1.
if AgeXmosyrsCat = 0 AgeXmosyrs = 0.
if AgeXmosyrsCat = 1 AgeXmosyrs = 1.
execute.

* Handle problem values.
*********************************
* If age at exit can't be calculated, recode to 9999 (missing DOB).
do if DQ_missDOB = 1.
recode AgeXmos AgeXyrs AgeXmosyrsCat AgeXmosyrs (else = 9999).
end if.

MISSING VALUES AgeXmos AgeXyrs AgeXmosyrsCat AgeXmosyrs  (7777, 8888,
9999).
execute.

* Formatting.
*************************************************

variable labels
EntryYr 'FFY child entered'
```

```
AgeNmos 'Age at entry in months'
AgeNyrs 'Age at entry in years (Under 1, 1, 2, 3 ... 21)'
AgeNmosyrs 'Age at entry in months (0-3 mo, 4-11 mo) and years (1, 2, 3
... 21)'
AgeNmosyrsCat 'Age at entry in months (0-3 mo, 4-11 mo) and years in
categories (1-5, 6-10, etc...)'
AgeXmos 'Age at exit in months'
AgeXyrs 'Age at exit in years (Under 1, 1, 2, 3 ... 21, 7777 for still in
care)'
AgeXmosyrs 'Age at exit in months (0-3 mo, 4-11 mo) and years (1, 2, 3
... 21, 7777 for still in care)'
AgeXmosyrsCat 'Age at exit in months (0-3 mo, 4-11 mo) and years in
categories (1-5, 6-10, ... 7777 for still in care)'.

* Create macro that holds age values.
DEFINE !AgeYrs ( )
0 '< 1 yr'
1 '1 yr'
2 '2 yrs'
3 '3 yrs'
4 '4 yrs'
5 '5 yrs'
6 '6 yrs'
7 '7 yrs'
8 '8 yrs'
9 '9 yrs'
10 '10 yrs'
11 '11 yrs'
12 '12 yrs'
13 '13 yrs'
14 '14 yrs'
15 '15 yrs'
16 '16 yrs'
17 '17 yrs'
18 '18 yrs'
19 '19 yrs'
20 '20 yrs'
21 '21 yrs'
!ENDDEFINE.

DEFINE !AgeMosYrs ( )
0 '0-3 mos'
1 '4-11 mos'
2 '1 yr'
3 '2 yrs'
4 '3 yrs'
5 '4 yrs'
6 '5 yrs'
7 '6 yrs'
8 '7 yrs'
9 '8 yrs'
10 '9 yrs'
11 '10 yrs'
12 '11 yrs'
```

```
13 '12 yrs'
14 '13 yrs'
15 '14 yrs'
16 '15 yrs'
17 '16 yrs'
18 '17 yrs'
19 '18 yrs'
20 '19 yrs'
21 '20 yrs'
22 '21 yrs'
!ENDDEFINE.

DEFINE !AgeMosYrsCat ( )
0 '0-3 mos'
1 '4-11 mos'
2 '1-5 yrs'
3 '6-10 yrs'
4 '11-16 yrs'
5 '17'
6 '18-21 yrs'
!ENDDEFINE.

* Age at entry.
value labels
AgeNmos
8888 'Invalid (age is negative or > 21)'
9999 'Cannot calculate (DOB or date of latest removal missing)'
/AgeNyrs
!AgeYrs
8888 'Invalid (age is negative or > 21)'
9999 'Cannot calculate (DOB or date of latest removal missing)'
/AgeNmosyrs
!AgeMosYrs
8888 'Invalid (age is negative or > 21)'
9999 'Cannot calculate (DOB or date of latest removal missing)'
/AgeNmosyrsCat
!AgeMosYrsCat
8888 'Invalid (age is negative or > 21)'
9999 'Cannot calculate (DOB or date of latest removal missing)'.

* Age at exit.
value labels
AgeXmos
7777 'Not applicable (child still in care as of the end of the 6-month
report)'
8888 'Invalid (age is negative or > 21)'
9999 'Cannot calculate (DOB or date of latest removal missing)'
/AgeXyrs
!AgeYrs
7777 'Not applicable (child still in care as of the end of the 6-month
report)'
8888 'Invalid (age is negative or > 21)'
9999 'Cannot calculate (DOB missing)'
/AgeXmosyrs
```

```
!AgeMosYrs
7777 'Not applicable (child still in care as of the end of the 6-month
report)'
8888 'Invalid (age is negative or > 21)'
9999 'Cannot calculate (DOB or date of latest removal missing)'
/ AgeXmosyrsCat
!AgeMosYrsCat
7777 'Not applicable (child still in care as of the end of the 6-month
report)'
8888 'Invalid (age is negative or > 21)'
9999 'Cannot calculate (DOB missing)'.

formats AgeNmos to AgeXmosyrs (F2.0).

* DisReason1
**************************************************

* Same as the original reason reported by the state, except when disreasn
is missing, uses 7777 for still in care.
* Note: The number of reports where DisReasn1 = 7777 will not equal the
number of reports where AgeX = 7777,
* because for DisReasn1 we define still in care as sysmis(disreasn) or
disreasn = 0, and for AgeX we define
* still in care as sysmis(DtDisch).
compute DisReason1 = disreasn.
if sysmis(disreasn) or disreasn = 0 DisReason1 = 7777.
execute.

MISSING VALUES DisReason1 (7777, 8888, 9999).

variable labels DisReason1 'Discharge reason (or 7777 if child still in
care)'.
value labels DisReason1
1 'Reunification'
2 'Living with a Relative'
3 'Adoption'
4 'Emancipation'
5 'Guardianship'
6 'Transfer to another agency'
7 'Runaway'
8 'Death of child'
7777 'Child was still in care as of the end of the 6-month report'.
formats DisReason1 (F4.0).

* Create DisReason2
**************************************************
* DisReason2 groups disreasns into four categories: Perm, NonPerm, Death,
or still in care.
recode DisReason1 (1,2,3,5=1) (4,6,7=2) (8=3) (7777=7777) into
DisReason2.
execute.

MISSING VALUES DisReason2  (7777, 8888, 9999).
```

```
variable labels DisReason2 'Discharge reason (perm, non perm, death) or
still in care (7777)'.
value labels DisReason2
1 'Permanency (all four types)'
2 'Non-Permanency'
3 'Child died'
7777 'Child was still in care as of the end of the 6-month report'.
formats DisReason2 (F4.0).

* Total number of removals (categorical)
************************************************
* not used for CFSR 3 performance, but useful for reporting and
descriptive statistics and context data.

recode totalrem (1=1) (2=2) (3=3) (4 thru Highest=4) (else=sysmis) into
TRemCat.
execute.

variable labels TRemCat 'Total number of removals from home
(categorical)'.
value labels TRemCat
1 '1'
2 '2'
3 '3'
4 '4 or more'.
formats TRemCat (F2.0).

*Race/ethnicity.

recode
amiakn asian blkafram hawaiipi white untodetm
(sysmis=0).
exe.
COUNT
  chrace = amiakn asian blkafram hawaiipi white untodetm  (1)  .
EXECUTE .
do if (chrace=1 and amiakn=1).
compute race=1.
else if (chrace=1 and asian=1).
compute race=2.
else if (chrace=1 and blkafram=1).
compute race=3.
else if (chrace=1 and hawaiipi=1).
compute race=4.
else if (chrace=1 and white=1).
compute race=6.
else if (chrace=1 and untodetm=1).
compute race=7.
else if (chrace>1).
compute race=8.
else if (chrace=0).
compute race=9.
end if.
formats race (f1.0).
```

```
value labels race
1'American Indian and Alaskan Native'
2'Asian'
3'Black or African American'
4'Native Hawaiian and Other Pacific Islander'
6'White'
7'Unable to Determine'
8'Two or More'
9'Missing'.
exe.
recode
hisorgin
(sysmis=0).
do if ((hisorgin=0)and (race=9)).
compute racethnc=9.
else if ((hisorgin=0 and race~=9)).
compute racethnc=race.
else if (hisorgin=1).
compute racethnc=5.
else if ((hisorgin=2) and (race~=9)).
compute racethnc=race.
else if ((hisorgin=2) and (race=9)).
compute racethnc=7.
else if ((hisorgin=3) and (race~=9)).
compute racethnc=race.
else if ((hisorgin=3) and (race=9)).
compute racethnc=7.
end if.
formats racethnc (f1.0).
value labels racethnc
1'American Indian/Alaskan Native-Non Hispanic'
2'Asian-Non Hispanic'
3'Black or African American-Non Hispanic'
4'Native Hawaiian/Other Pacific Islander-Non Hispanic'
5'Hispanic'
6'White-Non Hispanic'
7'Unknown/Unable to Determine'
8'Two or More-Non Hispanic'
9'Missing'.
exe.
recode racethnc (9=sysmis).
exe.

* Save.
**************************************************

save outfile='AFCARS_Comb_FH\CFSR 3 AFCARS source data.sav'
  /COMPRESSED.

* Save output.
output save OUTFILE='Root_FH\5 - SPSS Logs\CFSR 3 - Create AFCARS source
data.spv'.
```

CFSR 3: CREATE NCANDS SOURCE DATA

```
* Encoding: UTF-8.
* This code is designed to create an NCANDS source data file. The NCANDS
source datafile consists of multiple years of NCANDS child files with any
additional variables for analyses added at the end.
* The child files are combined using some python code that takes all the
files in a spefied

* Root folder for the data.
****PLEASE CHANGE THIS FILE PATH BASED ON WHERE YOU UNZIPED AND SAVED THE
CFSR 3 FOLDER.****.
file handle Root_FH /name="C:\CFSR 3".

***DO NOT CHANGE FILE HANDLES BELOW****.
*Folder where the NCANDS Submissions need for the SourceData file are
located.
file handle NCANDS_Source_FH  /name="Root_FH\0 - Source Data\Submissions
NCANDS".
*Folder for storing the combine child file and the combined DQ files.
file handle NCANDS_Comb_FH  /name="Root_FH\0 - Source Data".
*Forlder that contains fixed files used in the DQ process.

file handle Fixed_FH /name= "Root_FH\1 - Fixed files".

*******************************************************************
*******************************************************************
******************.
*Merge NCANDS Child Files.
*******************************************************************
*******************************************************************
******************.
*This is small set of python code designed to loop through the a folder
and merge all files that end with .sav. This code is run on the NCANDS
Source Folder
* (see: NCANDS_Source_FH) which is expected to contain all years of data
to be examine. The end result will be a data file with all years of
ncands to be analyzed.

begin program.
import spss, spssaux, os, glob

fh = spssaux.FileHandles()

rdir = fh.resolve('NCANDS_Source_FH')  # Specifies folder containing .sav
files.
fils = sorted([fil for fil in os.listdir(rdir) if fil.endswith('.sav')])
spss.Submit('get file "%s/%s".'%(rdir,fils.pop(0)))
for rep in range(len(fils)/49 + 1):
    spss.Submit('add files
file=*/%s.'%'/'.join(['file="%s"'%os.path.join(rdir,fil) for fil in
fils[49*rep:49*rep + min(49,len(fils)-49*rep)]]))
spss.Submit('exe.')
end program.

dataset name SourceFile_NCANDS.
```

```
DATASET ACTIVATE SourceFile_NCANDS.
rename variables staterr=stateabb.
ALTER TYPE stateabb (A6).
SORT CASES by stateabb subyr.

*****************************************************************************
*****************************************************************************
*******************.
*Compute Victim Flag and remove unborn children from the analyses.
*****************************************************************************
*****************************************************************************
*******************.

* Flag victims, where a victim is a child who died due to maltreatment
(maldeath = 1) or has a disposition of
* substantiated (malNlevel = 1) or indicated (malNlevel = 2), for any of
four possible maltreatments.
Compute flvictim = 0.
if ((mal1lev le 2) OR (mal2lev le 2) OR (mal3lev le 2) OR (mal4lev le 2)
OR (maldeath = 1)) flvictim = 1.

*Remove unborn children from analysis. Unborn children have a code of 77
in the NCANDS data.
select if chage ne 77.
execute.
missing values chage (99).

*****************************************************************************
*****************************************************************************
*******************.
* Merge in State name and state abbreviation from a separate file.
*****************************************************************************
*****************************************************************************
*******************.

get file='fixed_FH/states.sav'
/keep statetxt stateabb  state.
DATASET NAME STATES.
DATASET ACTIVATE STATES.

sort cases by stateabb.

match files
   /file = SourceFile_NCANDS
   /table = STATES
   /by stateabb.
execute.
DATASET Close States.

*****************************************************************************
*****************************************************************************
*******************.
* Save the NCANDS source data
```

```
**********************************************************************
**********************************************************************
******************.

dataset name SourceFileNCANDS.
DATASET Close SourceFile_NCANDS.
DATASET ACTIVATE SourceFileNCANDS.
SAVE OUTFILE = 'NCANDS_Comb_FH\CFSR 3 NCANDS source data.zsav'
/ZCOMPRESSED..
```

CFSR 3: CREATE AFCARS DATA QUALITY (DQ) FILES

```
* Encoding: UTF-8.

    ************************************************
    ************************************************
    CFSR 3 - CREATE AFCARS DQ FILES
    ************************************************
    ************************************************

    * Root folder for the data.
****PLEASE CHANGE THIS FILE PATH BASED ON WHERE YOU UNZIPED AND SAVED THE
CFSR 3 FOLDER.****.
    file handle Root_FH /name="C:\CFSR 3".

***DO NOT CHANGE FILE HANDLES BELOW****.
    * Folder where CFSR syntax is stored, in particular
"createDQFiles.sps" must be in the syntax folder.
    file handle Syntax_FH /name="Root_FH\0 - Syntax".
    *Folder for storing the combine sixmonth file and the combined DQ
files.
    file handle AFCARS_Comb_FH  /name="Root_FH\0 - Source Data".
    *Folder for the outputs of the split procedure related to running the
6month DQ checks.
    *The code will overwrite the entire contents of the Splits folder
each time it is run!.
    file handle splits_FH /name="Root_FH\0 - Temp files\Splits".
    *Folder where the individual DQ results are saved.
    file handle AFCARS_DQ_FH/name="Root_FH\2 - DQ AFCARS".
    *Folder where the SPSS logs are saved.
    file handle SPSSLogs_FH  /name="Root_FH\5 - SPSS Logs".
    * Folder where fixed files are located (fixed files are generally
useful static files that are used in various different sets of code).
    * FixedFiles must have "CFSR 3 AFCARS data quality checks.sav'".
    * FixedFiles must have "States.sav'" (a list of all states by state
number, state name, and state abbreviation).
    file handle Fixed_FH /name="Root_FH\1 - Fixed files".

    file handle TableauData_FH /name="Root_FH\6 - Data for Tableau".


*********************************************************************
****
    * Step 1: Split source file into separate sav files for each 6-month
period (i.e., DtReportEnd).

*********************************************************************
****

    * Specifies the location of the Splits folder, which much already
exist (referenced later in syntax as "splits").
    show handles.

    get file 'AFCARS_Comb_FH\CFSR 3 AFCARS source data.sav'.
    dataset name SourceData.
```

```
    * Create splitting variable with recognizable name for result files.
    string SixMoPeriod(A10).
    * Last two digits of the year.
    compute SixMoPeriod =
char.substr(string(xdate.year(DtReportEnd),F4.0), 3).
    do if xdate.month(DtReportEnd) eq 3.
      compute SixMoPeriod = concat(rtrim(SixMoPeriod), 'A').
    else.
      compute SixMoPeriod = concat(rtrim(SixMoPeriod), 'B').
    end if.
    execute.
    * Break dataset into one file for each 6-mo period and create a list
of the files created.
    * Note that anything in the target directory (i.e., Splits) wil be
deleted before execution!.
    SPSSINC SPLIT DATASET SPLITVAR=SixMoPeriod
    /OUTPUT DIRECTORY= "splits_FH" DELETECONTENTS=YES
    /OPTIONS NAMES=VALUES nameprefix="source"
    FILELIST="Splits_FH\splits.txt".

*************************************************************************
****
* Step 2: Create 6-month DQ files, iterating over each split source file.
*************************************************************************
****

* Note: This step calls the separate syntax file, CreateDQFiles.sps.,
which should be in ...\Syntax.

* The FILELIST parameter below uses the list of files created earlier by
SPSSINC SPLIT DATASET.
* If the list of files from that job needs to be changed to include a
different set of files, change FILELIST= below
* to a wildcard syntax INPUTDATA=“directory wildcard specification”,
* e.g., INPUTDATA="work\DQ AFCARS\splits\*.sav.

* PROCESS FILES iterates over a set of files and applies the SYNTAX file
to each one.
* The spv output is also saved in this directory, and a log file that
lists progress and any errors is also created
* or appended to in that directory.

SPSSINC PROCESS FILES  FILELIST="Splits_FH\splits.txt"
SYNTAX="Syntax_FH\createDQFiles.sps" OUTPUTDATADIR="AFCARS_DQ_FH"
CONTINUEONERROR=YES
VIEWERFILE= "SPSSLogs_FH\creation.spv" CLOSEDATA=YES
LOGFILE="SPSSLogs_FH\log.txt" MACRONAME="!JOB" LOGFILEMODE=APPEND.

*************************************************************************
****
* Step 3: Merge 6-month DQ files and run report.
*************************************************************************
****
```

```
* Close any existing data files or datasets.
dataset close all.
new file.

begin program.
import spss, spssaux, os, glob

fh = spssaux.FileHandles()

rdir = fh.resolve('AFCARS_DQ_FH')  # Specifies folder containing .sav
files.
fils = sorted(glob.glob(rdir + os.sep + "source_*.sav"))
spss.Submit('''get file "%s".
dataset name DQResults.'''  % (fils.pop(0)))
while True:
    adds = ['/file="%s"' % f for f in fils[:min(len(fils), 49)]]
    spss.Submit("""add files /file=*\n%s""" % "\n".join(adds))
    if len(fils) <=49:
        break
    fils = fils[49:]
end program.

*save data that will be used to eliminate states that fail DQ tests from
indicator analyses.
save outfile='AFCARS_DQ_FH\Merged AFCARS DQ files.sav'
  /COMPRESSED.

*save data that will be used for putting DQ results in State profiles.

save translate outfile = 'TableauData_FH\Merged AFCARS DQ.csv'
/FIELDNAMES
/cells=labels
/TYPE=CSV
/keep state stateabb SixMoPeriod DtReportBeg DtReportEnd DtReportEndFinal
TotalRecords DQVar ProblemRecords DQResult DQFileXW DQLimit DQCheckShort
DQCheckLong PermReEntry PS MALFC DQFailedCheck DQDirection
/replace.


* Save output.
begin program.
import os, re, spss

lastfile = os.path.basename(fils[-1]).split(".")[0]
unit = re.search(r"\d+[AB]", lastfile).group(0)    # identifier for last
period - suitable file must exist

spss.Submit(r"""output save OUTFILE='SPSSLogs_FH\CFSR 3 - Create AFCARS
DQ files up to %s.spv'.""" % unit)
end program.
```

CFSR 3: CREATE NCANDS DATA QUALITY (DQ) FILES

```
* Encoding: UTF-8.
********************************************************************************
********************************************************************************
*******************
NCANDS Data Quality Checks
********************************************************************************
********************************************************************************
*******************

* PURPOSE: The purpose of this code is to run the five NCANDS related DQ
checks. The 5 DQ checks are as following:
1) Child IDs for victims match across years
2) Child IDs for victims match across years, but dates of birth and sex
do not match
3) Missing age for victims
4) Some victims should have AFCARS IDs in Child File
5) Some victims with AFCARS IDs should match IDs in AFCARS Files

*More information on these checks can be found here:
_____________________________________________.

* REQUIREMENTS:
* This code assumes the previous code files have been run. In particular
a NCANDS source data file and a AFCARS source data file must exist.
********************************************************************************
********************************************************************************
*******************

********************************************************************************
********************************************************************************
*******************
Establish File Handles.
********************************************************************************
********************************************************************************
*******************.

*The file handle Root_FH represents the root folder for CFSR.
****PLEASE CHANGE THIS FILE PATH BASED ON WHERE YOU UNZIPED AND SAVED THE
CFSR 3 FOLDER.****.
file handle Root_FH /name="C:\CFSR 3".


***DO NOT CHANGE FILE HANDLES BELOW****.
*The file handle NCANDS_Comb_FH is an input folder for this code. It
should contain the combined NCANDS source files produced by earlier code.
file handle NCANDS_Comb_FH  /name="Root_FH\0 - Source Data".
*The file handle AFCARS_Comb_FH is an input folder for this code. It
should contain the combined AFCARS source files produced by earlier code.
file handle AFCARS_Comb_FH  /name="Root_FH\0 - Source Data".
* The file handle DQfiles_FH is the output folder for this code. It will
contain the results of the DQ checks.
file handle NCANDS_DQ_FH /name= "Root_FH\2 - DQ NCANDS".
*The file handle NCANDS_DQ_XLS_FH .
file handle TableauData_FH /name="Root_FH\6 - Data for Tableau".
```

```
*The file handle Fixed_FH is the folder for fixed files (aka static
files).
file handle Fixed_FH /name= "Root_FH\1 - Fixed files".
* The file handle SpssLogs_FH is the output folder for the SPSS logs for
this code.
file handle SpssLogs_FH /name="Root_FH\5 - SPSS Logs".
* The file handle Syntax_FH must contain the two "sub-program" code files
to run. These are: NCANDS_RecordLevel_DQ.sps and
NCANDS_ChildLevel_DQ.sps.
file handle Syntax_FH /name="Root_FH\0 - Syntax".
* The file handle will contain temporary files as a result of running the
code. This folder will have its contents erased each time this code is
run!.
file handle Splits_FH /name="Root_FH\0 - Temp files\Splits".

*******************************************************************
*******************************************************************
******************
* Load the Source dataset and subset it to those required records and
variables.
*******************************************************************
*******************************************************************
******************

* Close any existing data files or datasets.
new file.

GET file='NCANDS_Comb_FH\CFSR 3 NCANDS source data.zsav'
    /keep= subyr stateabb statetxt rptid chid chage chbdate chsex
afcarsid flvictim.
CACHE.

 DATASET NAME DQmainAll.
 DATASET ACTIVATE DQmainAll.

*Select only victims.
select if flvictim = 1.
execute.

*******************************************************************
*******************************************************************
******************.
* Create files needed to run NCANDS DQ checks for all NCANDS children
(part 1 of 2). One year of data per file is used for within year data
checks.
*******************************************************************
*******************************************************************
******************.

* Break dataset into one file for each year period and create a list of
the files created.
* Note that anything in the target directory (i.e., Splits) wil be
deleted before execution!.
SPSSINC SPLIT DATASET SPLITVAR=subyr
```

```
/OUTPUT DIRECTORY= "Splits_FH" DELETECONTENTS=YES
/OPTIONS NAMES=VALUES nameprefix="DQ_NCANDS_R_"
FILELIST="Splits_FH\splitsR.txt".

*****************************************************************************
*****************************************************************************
*******************.
* Create files needed to run NCANDS DQ checks for all NCANDS children
(part 2 of 2). Two years of data per file are used for cross year data
checks.
*****************************************************************************
*****************************************************************************
*******************.
DATASET ACTIVATE DQmainAll2.

*Program that creates a list of unique subyrs found in the datafile and
creates a dataset which contains 2 consecutive years for each subyr.
BEGIN PROGRAM.
import spss, spssdata, spssaux, os, glob
s = set()
data = spssdata.Spssdata(indexes=['subyr'])
for case in data:
        s.add(case.subyr)
yearlist = list(s)
yearlist.sort()
del data
for year in yearlist:
    if year == yearlist[-1]:
        spss.Submit("Compute lastyear = 1.")
    else:
        spss.Submit("Compute lastyear = 0.")
    spss.Submit("TEMPORARY.")
    spss.Submit("Select if subyr = " + str(year)+ " OR subyr = " +
str(year + 1) + ".")
    spss.Submit("Compute YearToUse = " + str(year) + ".")
    spss.Submit("Save outfile = 'Splits_FH\DQ_NCANDS_C_"+ str(int(year))
+ ".sav'.")
END PROGRAM.

*****************************************************************************
*****************************************************************************
*******************.
* Run the NCANDS DQ checks on the above generated  files using SPSSINC
process files.
*****************************************************************************
*****************************************************************************
*******************.

* Run DQ checks that are appropriate for record level/within year checks.
SPSSINC PROCESS FILES  FILELIST="Splits_FH\splitsR.txt"
SYNTAX="Syntax_FH\NCANDS_RecordLevel_DQ.sps" OUTPUTDATADIR="NCANDS_DQ_FH"
CONTINUEONERROR=YES
VIEWERFILE= "SpssLogs_FH\NCANDS_DQ_R_creation.spv" CLOSEDATA=YES
```

```
LOGFILE="SpssLogs_FH\NCANDS_DQ_R_log.txt" MACRONAME="!JOB"
LOGFILEMODE=APPEND.

* Run DQ checks that are appropriate for child level /between year
checks.
SPSSINC PROCESS FILES  INPUTDATA="Splits_FH\DQ_NCANDS_C_*.sav"
SYNTAX="Syntax_FH\NCANDS_ChildLevel_DQ.sps" OUTPUTDATADIR="NCANDS_DQ_FH"
CONTINUEONERROR=YES
VIEWERFILE= "SpssLogs_FH\NCANDS_DQ_C_creation.spv" CLOSEDATA=YES
LOGFILE="SpssLogs_FH\NCANDS_DQ_C_log.txt" MACRONAME="!JOB"
LOGFILEMODE=APPEND.

*************************************************************************
*************************************************************************
*******************.
* Combine the state/year level results of the DQ checks into a single
file.
*************************************************************************
*************************************************************************
*******************.

begin program.
import spss, spssaux, os, glob

fh = spssaux.FileHandles()

rdir = fh.resolve('NCANDS_DQ_FH')  # Specifies folder containing .sav
files.
fils = sorted(glob.glob(rdir + os.sep + "DQ_NCANDS_*.sav"))
spss.Submit('''get file "%s".
dataset name DQResults.'''  % (fils.pop(0)))
while True:
    adds = ['/file="%s"' % f for f in fils[:min(len(fils), 49)]]
    spss.Submit("""add files /file=*\n%s""" % "\n".join(adds))
    if len(fils) <=49:
        break
    fils = fils[49:]
end program.
 EXECUTE.

*************************************************************************
*************************************************************************
*******************.
* Merge the results with the fixed file NCANDS data quality checks.sav
which contains threshold and other variables required to calculate DQ
test pass/fail.
*************************************************************************
*************************************************************************
*******************.

Get file='Fixed_FH\CFSR 3 NCANDS data quality checks.sav'.
sort cases by DQVAR.
Save outfile='Fixed_FH\CFSR 3 NCANDS data quality checks.sav'.
```

```
DATASET ACTIVATE DQResults.
SORT CASES by DQVAR.
 EXECUTE.

match files
    /FILE=DQResults
    /TABLE='Fixed_FH\CFSR 3 NCANDS data quality checks.sav'
    /BY DQVar.
execute.
Dataset name DQRandI.
Dataset close DQResults.

COMPUTE DQFailedCheck=0.
DO IF DQDirection = "<".
    if DQResult < DQLimit DQFailedCheck=1.
ELSE IF DQDirection = ">".
    if DQResult > DQLimit DQFailedCheck=1.
ELSE if Upper(DQDirection) = "N".
    if DQResult = 0 DQFailedCheck=1.
END IF.

STRING Period (A12).
IF DQFileXW = "Within file" Period = STRING(Subyr,F4.0).
IF (DQFileXW = "Cross file" AND DQVAR = "AFCARSMatch") Period =
STRING(Subyr,F4.0).
IF (DQFileXW = "Cross file" AND DQVAR ~= "AFCARSMatch") Period =
CONCAT(STRING(Subyr,F4.0), "-", STRING(Subyr+1,F4.0)).

save outfile='NCANDS_DQ_FH\Merged NCANDS DQ files.sav'
  /COMPRESSED
  /keep statetxt stateabb subyr DQVAR DQSum DQN DQResult DQFailedCheck
Period DQFileXW DQDirection DQLimit DQCheckShort DQCheckLong MALFC
MALREC.

String MatchField (A8).
Compute MatchField = "".

save translate outfile = 'TableauData_FH\Merged NCANDS DQ.csv'
/FIELDNAMES
/rename = (statetxt = state)
/TYPE=CSV
/keep state DQVAR stateabb DQResult DQFailedCheck Period DQFileXW
DQDirection DQLimit DQCheckShort DQCheckLong MALFC MALREC MatchField
/replace.


GET FILE ='NCANDS_DQ_FH\Merged NCANDS DQ files.sav'.

DATASET CLOSE DQRandI.

**************************************************************************
**************************************************************************
******************.
```

```
* Combine the individual child/year level results of the DQ checks into a
single file.
*****************************************************************
*****************************************************************
******************.

begin program.
import spss, spssaux, os, glob

fh = spssaux.FileHandles()

rdir = fh.resolve('NCANDS_DQ_FH')  # Specifies folder containing .sav
files.
fils = sorted(glob.glob(rdir + os.sep + "IND_DQ_NCANDS_*.sav"))
spss.Submit('''get file "%s".
dataset name INDResults.'''  % (fils.pop(0)))
while True:
    adds = ['/file="%s"' % f for f in fils[:min(len(fils), 49)]]
    spss.Submit("""add files /file=*\n%s""" % "\n".join(adds))
    if len(fils) <=49:
        break
    fils = fils[49:]
end program.

DATASET ACTIVATE INDResults.

save outfile='NCANDS_DQ_FH\Merged NCANDS IND files.sav'
  /COMPRESSED.

NEW FILE.
DATASET CLOSE INDResults.
```

CFSR 3: OBSERVED PERFORMANCE FOR MALTREATMENT IN FOSTER CARE

```
* Encoding: UTF-8.
************************************
CFSR 3 Observed Performance for Maltreatment in Care
************************************

*The file handle Root_FH represents the root folder for the CFSR data.
****PLEASE CHANGE THIS FILE PATH BASED ON WHERE YOU UNZIP AND SAVE THE
CFSR 3 FOLDER.****.
file handle Root_FH /name="C:\CFSR 3".
*The file handle NCANDS_Comb_FH is the input folder for this code. It
should contain the combined NCANDS source files produced by earlier code.
file handle NCANDS_Comb_FH  /name="Root_FH\0 - Source Data".
*The file handle AFCARS_Comb_FH is the input folder for this code. It
should contain the combined AFCARS source files produced by earlier code.
file handle AFCARS_Comb_FH  /name="Root_FH\0 - Source Data".
file handle NCANDS_DQ_FH /name= "Root_FH\2 - DQ NCANDS".
file handle AFCARS_DQ_FH /name= "Root_FH\2 - DQ AFCARS".
file handle SpssLogs_FH /name="Root_FH\5 - SPSS Logs".
file handle Syntax_FH /name="Root_FH\0 - Syntax".
file handle Temp_FH /name= "Root_FH\0 - Temp files".
file handle Splits_FH /name="Temp_FH\Splits".
file handle Output_Child_FH /name="Root_FH\3 - Performance observed
child".
file handle Output_State_FH /name="Root_FH\3 - Performance observed
state".


* Open AFCARS source data.
get file 'root_FH\0 - Source Data\CFSR 3 AFCARS source data.sav'
/keep state fipscode recnumbr sex totalrem numplep curplset disreasn
ChildID statetxt stateabb DtReportBeg DtReportEnd DtReportEndFinal
DtBirth
DtFirRem DtPriorDisch DtLatRem DtCurSet DtDisch FY afcarsid AgeNmos
AgeNmosyrsCat AgeNyrs AgeNmosyrs  ExAge18 Bday18 Adjust18
DtDischAdjusted DQ_Dropped DQ_IDNoMatchNext6Mo DQ_missDOB DQ_missDtLatRem
DQ_missNumPlep DQ_DOBgtDtLatRem DQ_DOBgtDtDisch DQ_DtDischeqDtLatRem
DQ_DtDischltDtLatRem DQ_missDisreasn DQ_totalrem1 DQ_gt21DOBtoDtLatRem
DQ_gt21DOBtoDtDisch DQ_gt21DtDischtoDtLatRem DQ_dtpriordafterdlm
EntryYr ExitYr AgeXmos AgeXmosyrsCat AgeXyrs AgeXmosyrs TRemCat chrace
race racethnc.
dataset name Indicator window=front.
cache.


************************************************
* SPECIFY THE 12-MONTH COHORT WHOSE OUTCOME WILL BE ASSESSED *****MANUAL
INPUT REQUIRED HERE******
************************************************

* Identify the 12-month cohort of interest (e.g., children in care during
13A13B) by entering "AB" or "BA" for
* PeriodType and the year the 12-month period ends (for YYYY and YYstr).
```

```
* AB period (A file + B file) spans Oct 1 - Sept 30 of the following
year.
* BA period (B file + A file) spans Apr 1 - Mar 31 of the following year.

* Examples:
* ( * ) represents a FY
* Children in care during   PeriodType  YYYY   YYstr
* 09B10A                        BA              2010     10
* 10A10B *                      AB              2010     10
* 10B11A                        BA              2011     11
* 11A11B *                      AB              2011     11
* 11B12A                        BA              2012     12
* 12A12B *                      AB              2012     12
* 12B13A                        BA              2013     13
* 13A13B *                      AB              2013     13
* 13B14A                        BA              2014     14
* etc.

********  START USER INPUT ********

define PeriodType () "XX"
!enddefine.
define YYYY () XXXX.
!enddefine.
define YYstr () "XX"
!enddefine.

********  END USER INPUT ********

* Create variable that indicates the user-specified 12-month cohort
(e.g., "AB13" represents "13A13B").
string TwelveMoCohort (A4).
compute TwelveMoCohort = concat(PeriodType,YYstr).
execute.

************************************************
CREATE DATE PARAMETERS
************************************************

* Start date of a 12-month period (10/1/YYYY or 4/1/YYYY).
* End date of a 12-month period (3/31/YYYY or 9/30/YYYY).

* Maltreatment in foster care involves following children who were served
during the 12-month period specified
* earlier. Served includes children who entered care during the period,
or were in care on the first day of the
* period. Identify the start date (DtPeriodBeg) and end date
(DtPeriodEnd) that defines this 12-month period.
*ALL ANALYSIS PERIODS ARE AB FOR MALTREATMENT IN CARE.

compute DtPeriodBeg=date.mdy(10,01,YYYY-1).
compute DtPeriodEnd=date.mdy(09,30,YYYY).
compute DtEndFirst6mo=date.mdy(03,31,YYYY).
EXECUTE.
```

```
formats DtPeriodBeg DtPeriodEnd DtEndFirst6mo(adate10).

variable labels
TwelveMoCohort '12-month period children were served and for whom
performance is being assessed'
DtPeriodBeg 'Start date of the 12-month period specified'
DtPeriodEnd 'End date of the 12-month period specified'.

************************************************
* REMOVE STATES THAT EXCEED DQ LIMITS FOR THE AFCARS DQ CHECKS
************************************************

* The AFCARS cross-file checks (Dropped records and AFCARS IDs don't
match from one period to the next)
* apply only to the first 6-month period (DtReportBeg = DtPeriodBeg). The
AFCARS within-file checks apply to
* both periods (DtReportBeg ge DtPeriodBeg AND DtReportBeg le
DtPeriodEnd). The NCANDS DQ checks
* will be applied later.

* Open AFCARS data quality results.
get file 'AFCARS_DQ_FH\Merged AFCARS DQ files.sav'.
dataset name DQResults window=front.

* Prevent accidentally overwriting file.
dataset copy  DQIndicator.
dataset activate DQIndicator.
dataset close DQResults.

do if PeriodType = "AB".
compute DtPeriodBeg=date.mdy(10,01,YYYY-1).
compute DtPeriodEnd=date.mdy(09,30,YYYY).
end if.

execute.
formats DtPeriodBeg DtPeriodEnd (adate10).

variable labels
DtPeriodBeg 'Start date of the 12-month period specified'
DtPeriodEnd 'End date of the 12-month period specified'.


* Select the AFCARS checks for this indicator and DQ results for the 6-
month period(s).
* ... Cross file checks (apply only to the first 6-month period).
if MALFC = 1 and DQFileXW = "Cross file" AND (DtReportBeg eq DtPeriodBeg)
DQChecksApply = 1.
* ... Within file checks (apply to both 6-month periods).
if MALFC = 1 and DQFileXW = "Within file" AND (DtReportBeg ge
DtPeriodBeg) and (DtReportEnd le DtPeriodEnd) DQChecksApply = 1.
execute.

* For each state, for the checks that apply, sum the number of checks it
failed.
```

```
select if DQChecksApply = 1.
execute.


DATASET DECLARE DQStateInd.
aggregate
  /OUTFILE=DQStateInd
  /BREAK=stateabb
  /DQFailedCheck_sum=SUM(DQFailedCheck).
dataset activate DQStateInd.
dataset close DQIndicator.
sort cases by stateabb.

get file 'NCANDS_DQ_FH\Merged NCANDS DQ files.sav'.
dataset name DQResultsN window=front.
cache.
* Prevent accidentally overwriting file.
dataset copy  DQIndicatorN.
dataset activate DQIndicatorN.
dataset close DQResultsN.


if ((malFC eq 1) and (subyr ge YYYY) AND (subyr le (YYYY+1)))
DQchecksApplyN =1.
sort cases by stateabb.
select if DQChecksApplyn = 1.
execute.


DATASET DECLARE DQIndicatorN2.
aggregate
  /OUTFILE=DQIndicatorN2
  /BREAK=stateabb
  /DQFailedCheckN_sum=SUM(DQFailedCheck).

match files
    /FILE='DQStateInd'
    /FILE='DQIndicatorN2'
    /BY stateabb.
execute.

dataset name DQIndicators.
dataset activate DQIndicators.
dataset close DQStateInd.
dataset close DQIndicatorN.
dataset close DQIndicatorN2.

sort cases by stateabb.
* ... Merge in all variables from DQIndicator file, then keep only
DQFailedCheck_sum.
dataset activate Indicator.
sort cases by stateabb.
execute.
match files
```

```
    /FILE=*
    /TABLE='DQIndicators'
    /BY stateabb
    /KEEP state to DQFailedCheckN_sum.
execute.

dataset close DQIndicators.

* Select only states that met the DQ checks.
select if DQFailedCheck_sum eq 0 AND DQFailedCheckN_sum eq 0.
execute.
delete variables DQFailedCheck_sum DQFailedCheckN_sum.

if DtReportEnd eq DtPeriodEnd DQ_Dropped = 0.
execute.

* Flag records with a problem (i.e., = 1) for any of the AFCARS DQ checks
used for Maltreatment in foster care,
* except DQ_IDNoMatchNext6Mo and DQ_totalrem1. The NCANDS DQ checks will
be applied later.
compute DQ_Indicator=0.
if any(1,DQ_DOBgtDtDisch, DQ_DOBgtDtLatRem, DQ_Dropped,
DQ_DtDischeqDtLatRem, DQ_DtDischltDtLatRem, DQ_gt21DOBtoDtDisch,
DQ_gt21DOBtoDtLatRem, DQ_gt21DtDischtoDtLatRem, DQ_missDOB,
DQ_missDtLatRem, DQ_dtpriordafterdlm) DQ_Indicator=1.
execute.

select if DQ_Indicator=0.
execute.

*Select only records reported during the period of interest for the
analysis.
if (DtReportBeg ge DtPeriodBeg) and (DtReportEnd le DtPeriodEnd)
ReportedDuringPeriod = 1.
execute.

select if ReportedDuringPeriod=1.
execute.
************************************************

* For maltreatment in foster care, all of a child's episodes during the
12-month period are considered.
* For example, if a child has two entries in the 12-month period, data
from both episodes are used.

* Select only the most recent 6-month record reported for each child, for
each episode.
************************************************

* This will create an episode-level file. Episodes are distinguished by
the date of latest removal from home.
* A child has a record in each 6-month submission until he discharged,
dropped, or was still in care as of
```

```
* the last reporting period we have in the file. We only want one record
per child, per episode, and the record we
* pick is the last one reported for that episode. This record will
contain the most recent data available that
* describes the child's episode (e.g., LOS, age, etc.).

* Flag the records to keep.
sort cases BY ChildID (A) DtLatRem (A).
match files
  /FILE=*
  /BY ChildID DtLatRem
  /LAST=Flag_LastRpt4Ep.
variable labels Flag_LastRpt4Ep 'last 6-month report we received for this
episode (based on DtLatRem)'.
value labels Flag_LastRpt4Ep 0 'Duplicate Case' 1 'Primary Case'.
variable level Flag_LastRpt4Ep (ORDINAL).
execute.

* Select the last 6-month record for each child, for each episode.
select if Flag_LastRpt4Ep=1.
execute.

delete variables Flag_LastRpt4Ep.



* Create InAtStart, Entered, and Served Variables. Select only children
who were served during the 12-month period specified earlier.
**************************************************

* This is the cohort of children for whom performance will be examined.

if ((DtLatRem lt DtPeriodBeg) and (DtDisch ge (DtPeriodBeg) |
missing(DtDisch))) InAtStart = 1.
if ((DtLatRem ge DtPeriodBeg) and (DtLatRem le DtPeriodEnd)) Entered = 1.
if (InAtStart = 1 or Entered = 1) Served = 1.
recode InAtStart Entered (sysmis = 0).
execute.

variable labels
InAtStart 'Child was in care on the first day of the 12-month period'
Entered 'Child entered care during the specified 12-month period'.
value labels InAtStart Entered
0 'No'
1 'Yes'.
execute.

* Select the records to keep.
select if Served = 1.
execute.

**************************************************
* ADJUST DISCHARGE DATES FOR YOUTH WHO TURN 18
**************************************************
```

```
* Youth who turn 18 during the 12-month period will not have time in care
beyond their 18th birthday or
* victimizations after their 18th birthday counted. To handle this, for
children who turn 18 during the period we
* replace their discharge date with the date they turned 18. Therefore,
the LOS for children who turn 18 during
* the period will be calculated from date of latest removal to date of
18th birthday.
*************************************************

* Calculate child's 18th birthday.
compute Bday18=DATESUM(DtBirth, 18, "years", 'closest').
variable labels  Bday18 "18th birthday".
variable level Bday18(SCALE).
formats Bday18(ADATE10).
variable width Bday18(10).
execute.

* If 18th birthday occurred during the 12-month period, and child had not
discharged by the time he turned 18,
* replace date of discharge with date of 18th birthday.

do if (Bday18 ge DtPeriodBeg and Bday18 le DtPeriodEnd) and
(missing(DtDisch) or DtDisch gt Bday18).
compute DtDischAdjusted=Bday18.
compute Adjust18=1.
end if.
execute.
variable labels DtDischAdjusted 'For children who turned 18 during the
period and had not discharged by that time, the date of their 18th
birthday is their effective date of discharge'.

* Copy date of discharge for remaining children to DtDischAdjusted. This
will put the date of discharge that we want to use for every child in the
DtDischAdjusted variable.
do if sysmis(DtDischAdjusted).
compute DtDischAdjusted=DtDisch.
end if.
execute.
formats DtDischAdjusted (adate10).

*************************************************
* CALCULATE AGE ON FIRST DAY
*************************************************

* If child entered during the 12-month period, we use age entry (already
calculated and part of source data).
* If the child was in care on the first day of the 12-month period, we
use age on first day.

* Age on first day (categorical), specified as 0-3 mos, 4-11 mos, 1-5
yrs, 6-10 yrs, 11-16 yrs, 17, 18-21 yrs.
* Useful for reporting and descriptive statistics.
```

```
*********************************

* Calculate age on first day in months.
compute AgeFDmos = datediff(DtPeriodBeg,DtBirth,"months").
* Record age in months to desired categories.
recode AgeFDmos (216 thru 263=6) (204 thru 216=5) (132 thru 204=4) (72
thru 132=3) (12 thru 72=2) (4 thru 12=1) (0 thru 4=0) (else=sysmis) into
AgeFDmosyrsCat.
execute.

* Age on first day (interval), specified as 0-3 mos, 4-11 mos, 1 yr ...
21 yrs.
* Used later in risk adjustment, where each age value is converted to a
yes/no (1/0) dummy variable.
*********************************

* Calculate age on first day in years.
compute AgeFDyrs = datediff(DtPeriodBeg,DtBirth,"years").
* Children < 1 currently have a value of 0 for AgeFDyrs. Need to split
these into 0-3 mos (0) mos and 4-11 mos (1).
* To make room for the 0 and 1 for < 1 children, make a copy of AgeFDyrs
then shift all subsequent ages, so age1 = 2, age2 = 3, etc.
compute AgeFDmosyrs = AgeFDyrs.
* Recode 1 - 21 to 2 - 22.
recode AgeFDyrs (1=2) (2=3) (3=4) (4=5) (5=6) (6=7) (7=8) (8=9) (9=10)
(10=11) (11=12) (12=13) (13=14) (14=15) (15=16) (16=17) (17=18) (18=19)
(19=20) (20=21) (21=22) INTO AgeFDmosyrs.
* If child is 0-3, make AgeFDmosyrs = 0.
* If child is 4-11, make AgeFDmosyrs = 1.
if AgeFDmosyrsCat = 0 AgeFDmosyrs = 0.
if AgeFDmosyrsCat = 1 AgeFDmosyrs = 1.
execute.

* Handle problem values.
*********************************

* If date of birth is after the date of first day (i.e., age on FS is
negative), recode to 8888.
* Note: Negative age on first day is to be expected for children who
entered during the 12-month period at age
* < 1. For these children, their age at entry will be selected so it's
okay to assign their age on first day to 8888.
do if DtBirth > DtPeriodBeg.
recode AgeFDmos AgeFDyrs AgeFDmosyrsCat AgeFDmosyrs (else = 8888).
end if.
* If time between date of birth and date of first day is > 21 yrs (i.e.,
age on FD is > 21 yrs), recode to 8888.
compute #AgeFD = datediff(DtPeriodBeg,DtBirth,"years").
do if #AgeFD > 21.
recode AgeFDmos AgeFDyrs AgeFDmosyrsCat AgeFDmosyrs (else=8888).
end if.
execute.

variable labels
```

```
AgeFDmos 'Age on first day in months'
AgeFDyrs 'Age on first day in years (Under 1, 1, 2, 3 ... 21)'
AgeFDmosyrs 'Age on first day in months (0-3 mo, 4-11 mo) and years (1,
2, 3 ... 21)'
AgeFDmosyrsCat 'Age on first day in months (0-3 mo, 4-11 mo) and years in
categories (1-5, 6-10, etc...)'.

* Create macro that holds age values.
DEFINE !AgeYrs ( )
0 '< 1 yr'
1 '1 yr'
2 '2 yrs'
3 '3 yrs'
4 '4 yrs'
5 '5 yrs'
6 '6 yrs'
7 '7 yrs'
8 '8 yrs'
9 '9 yrs'
10 '10 yrs'
11 '11 yrs'
12 '12 yrs'
13 '13 yrs'
14 '14 yrs'
15 '15 yrs'
16 '16 yrs'
17 '17 yrs'
18 '18 yrs'
19 '19 yrs'
20 '20 yrs'
21 '21 yrs'
!ENDDEFINE.

DEFINE !AgeMosYrs ( )
0 '0-3 mos'
1 '4-11 mos'
2 '1 yr'
3 '2 yrs'
4 '3 yrs'
5 '4 yrs'
6 '5 yrs'
7 '6 yrs'
8 '7 yrs'
9 '8 yrs'
10 '9 yrs'
11 '10 yrs'
12 '11 yrs'
13 '12 yrs'
14 '13 yrs'
15 '14 yrs'
16 '15 yrs'
17 '16 yrs'
18 '17 yrs'
19 '18 yrs'
```

```
20 '19 yrs'
21 '20 yrs'
22 '21 yrs'
!ENDDEFINE.

DEFINE !AgeMosYrsCat ( )
0 '0-3 mos'
1 '4-11 mos'
2 '1-5 yrs'
3 '6-10 yrs'
4 '11-16 yrs'
5 '17'
6 '18-21 yrs'
!ENDDEFINE.

* Age on first day.
value labels
AgeFDmos
8888 'Invalid (age is negative or > 21)'
9999 'Cannot calculate (DOB missing)'
/AgeFDyrs
!AgeYrs
8888 'Invalid (age is negative or > 21)'
9999 'Cannot calculate (DOB missing)'
/AgeFDmosyrs
!AgeMosYrs
8888 'Invalid (age is negative or > 21)'
9999 'Cannot calculate (DOB missing)'
/AgeFDmosyrsCat
!AgeMosYrsCat
8888 'Invalid (age is negative or > 21)'
9999 'Cannot calculate (DOB missing)'.

formats AgeFDmos to AgeFDmosyrs (F2.0).

************************************************
* SET CHILD'S AGE AS AGE AT ENTRY OR AGE ON FIRST DAY
************************************************

* If child entered during the 12-month period, set age to age at entry.
* If child was in care on the first day of the 12-month period, set age
to age on first day.

do if Entered=1.
compute Agemos=AgeNmos.
compute AgemosyrsCat=AgeNmosyrsCat.
compute Ageyrs=AgeNyrs.
compute Agemosyrs=AgeNmosyrs.
end if.
do if InAtStart=1.
compute Agemos=AgeFDmos.
compute AgemosyrsCat=AgeFDmosyrsCat.
compute Ageyrs=AgeFDyrs.
compute Agemosyrs=AgeFDmosyrs.
```

```
end if.
execute.

variable labels
Agemos 'Age on first day (or at entry) in months'
Ageyrs 'Age on first day (or at entry) in years (Under 1, 1, 2, 3 ...
21)'
Agemosyrs 'Age on first day (or at entry) in months (0-3 mo, 4-11 mo) and
years (1, 2, 3 ... 21)'
AgeNmosyrsCat 'Age on first (or at entry) in months (0-3 mo, 4-11 mo) and
years in categories (1-5, 6-10, etc...)'.

value labels
Agemos
8888 'Invalid (age is negative or > 21)'
/Ageyrs
!AgeYrs
8888 'Invalid (age is negative or > 21)'
/Agemosyrs
!AgeMosYrs
8888 'Invalid (age is negative or > 21)'
/AgemosyrsCat
!AgeMosYrsCat
8888 'Invalid (age is negative or > 21)'.

formats Agemos to Agemosyrs (F2.0).

*************************************************
* CALCULATE LENGTH OF STAY
*************************************************

* LOS calculations differ based on when the children was in care (i.e.,
on the first day of or entered during the
* 12-month period) and when the child exited (i.e., during the 12-month
period or after, if at all).

* We already have a flag to indicate if the child was InAtStart or
Entered. Now create a flag to indicate if the
* child exited during the 12-month period.

*SPECIAL CALCULATIONS AND CONSIDERATIONS FOR CHILDREN WITH MORE THAN ONE
EPISODE IN THE 12-MONTH PERIOD OF INTEREST.

*Flag any episode that is not a child's first episode of the year.
sort cases by ChildID dtreportbeg.
if (childid eq lag(childid)) duplicate_child=1.
execute.

*Label each episode to indicate if it is a child's first or second
episode.
compute position=1.
if (duplicate_child=1) position = lag(position)+1.
execute.
```

```
if missing(duplicate_child) duplicate_child eq 0.
execute.

*create child-level flag for children with multiple epiosdes.

aggregate
outfile=* mode=ADDVARIABLES
/break state childid
/dupchild = max(duplicate_child).


*For each record take the dtpriordisch from the subquent episode and
create variable to hold it.

PRESERVE.
set workspace 200000.
create TempDtPriorDisch=Lead(DtPriorDisch,1).
execute.
RESTORE.

*For each episode, take total number of removals from next episode and
creat variable to hold it.
PRESERVE.
set workspace 200000.
create nexttotrem=Lead(totalrem,1).
execute.
RESTORE.

*Compute the change in the total number of removals between reported
episodes for children who have multiple episodes.
if (dupchild=1 and position=1) changerem = (nexttotrem - totalrem).
execute.


compute ExMissingDOD=0.
compute ExMissingPriorDOD=0.
EXECUTE.

*Set  DtDischNextRecord as TempDtPriorDisch in first record if child has
more than one episode, total number of removals only increased by 1, and
the tempdtpriordisch occurs after the date of removal for the first
episode reported in the period.
*If child turns 18, only set  DtDischNextRecord as TempDtPriorDisch if it
occured before the child's 18th birthday.
*Change DtDischAdjusted to DtDischNextRecord where applicable.

if dupchild=1 and position=1 and adjust18 eq 1 and changerem eq 1 and
tempdtpriordisch ge dtlatrem and (TempDtPriorDisch lt DtDischAdjusted)
AND (DtDischAdjusted eq Bday18) DtDischNextRecord = TempDtPriorDisch.
if dupchild=1 and position=1 and missing(DtDischAdjusted) AND changerem
eq 1 and tempdtpriordisch ge dtlatrem and adjust18 ne 1
DtDischNextRecord=TempDtPriorDisch.
execute.
```

```
if (DtDischNextRecord ge dtperiodbeg and DtDischNextRecord le
DtPeriodEnd) DtDischAdjusted=DtDischNextRecord.
execute.


*Drop first record if child has more than one episode, the number of
removals between the first record and the second record changes by more
than one, and we do not have a valid dtdisch reported.
compute drop_changerem =0.
if dupchild eq 1 and position eq 1 and changerem ne 1 and
missing(dtdisch) drop_changerem = 1.
execute.
select if drop_changerem ne 1.
execute.


************************************************
* CALCULATE LENGTH OF STAY
************************************************

* LOS calculations differ based on when the children was in care (i.e.,
on the first day of or entered during the
* 12-month period) and when the child exited (i.e., during the 12-month
period or after, if at all).

* We already have a flag to indicate if the child was InAtStart or
Entered. Now create a flag to indicate if the child exited during the 12-
month period.
if ((DtDischAdjusted ge DtPeriodBeg) and (DtDischAdjusted le
DtPeriodEnd)) Exited = 1.
recode Exited (sysmis = 0).
execute.
variable labels Exited 'Child exited care during the specified 12-month
period'.
value labels Exited
0 'No'
1 'Yes'.
execute.

compute LOSdays = $sysmis.
* If child entered and exited during the 12-month period, LOS is time
between DtLatRem and DtDischAdjusted.
do if Entered=1 and Exited=1.
compute LOSdays=datediff(DtDischAdjusted,DtLatRem,"days").
end if.
* If child was in care at the start of the 12-month period, and exited
during the 12-month period, LOS is time
* between DtPeriodBeg and DtDischAdjusted.
do if InAtStart=1 and Exited=1.
compute LOSdays=datediff(DtDischAdjusted,DtPeriodBeg,"days").
end if.
* If child entered during the 12-month period, and did not exit during
the 12-month period, LOS is time between
* DtLatRem to DtPeriodEnd.
```

```
do if Entered=1 and Exited=0.
compute LOSdays=datediff(DtPeriodEnd,DtLatRem,"days").
end if.
* If child was in care at the start of the 12-month period, and did not
exit during the 12-month period, LOS is
* time between DtPeriodBeg and DtPeriodEnd. These children were in care
continuously across the entire year,
* so we add a day to their LOS. Otherwise, SPSS does not count inclusive
of the start and end dates.
do if InAtStart=1 and Exited=0.
compute LOSdays=datediff(DtPeriodEnd,DtPeriodBeg,"days").
compute LOSdays=LOSdays+1.
end if.
execute.

variable labels
LOSdays 'LOS - Days in care during the 12-month period'.
formats LOSdays (F4.0).

****Remove records that do not have a valid discharge date reported in
dtdischadjusted or dtpriordisch from next record for children with 2
records that have different dtlatrem.

sort cases by ChildID dtreportbeg.
if (dtlatrem ne lag(dtlatrem) AND childID eq lag(ChildID) and
(missing(lag(dtdischadjusted)) OR lag(Adjust18) eq 1)) flagCase eq 1.
execute.

if flagcase eq 1 AND ((missing(dtpriordisch)) or dtpriordisch lt
dtperiodbeg) flagcase2 eq 1.
execute.

aggregate
  /OUTFILE=* MODE=ADDVARIABLES
  /BREAK=ChildID
  /Flagrptremove2 = max(flagcase2).


compute dropreport eq 0.
if (flagrptremove2 eq 1 AND position eq 1) dropreport eq 1.
execute.


select if dropreport ne 1.
execute.

***DROP first record/episode if multiple records/episodes are reported
for a child and they contain conflicting/overlapping information.

if childid eq lag(childid) AND dtlatrem lt lag(dtdisch) AND
not(missing(lag(dtdisch))) overlapep eq 1.
execute.
```

```
aggregate
  /OUTFILE=* MODE=ADDVARIABLES
  /BREAK=ChildID
  /overlapepchild = max(overlapep).

compute dropoverlap eq 0.
if (overlapepchild eq 1 AND position eq 1) dropoverlap = 1.
execute.

select if dropoverlap eq 0.
execute.

*************************************************
* REMOVE EPISODES THAT MEET EXCLUSION CRITERIA
*************************************************

* The denominator for this indicator is the sum of children's LOS (days)
across *all* episodes during the
* 12-month period. So if a child entered twice during the 12-month
periods, his total LOS is the LOS from his
* first episode plus the LOS from his second episode. However, when
summing a child's LOS over all his
* episodes in the 12-month period, we want to exclude:
* 1) episodes in which the child entered at age 18 or more and
* 2) *complete* episodes with LOS < 8 days. If the episode has a LOS < 8
days, but the child is still in care,
* we want to keep this. These short but ongoing episodes represent
entries that occurred near the end of the
* 12-month period and continued past it (i.e., child was still in care).

* Complete episodes with LOS < 8 days.
compute ExLOS8 = 0.
if (Entered=1 and Exited=1) and LOSdays lt 8 ExLOS8=1.
do if InAtStart=1 and Exited=1.
compute TempLOS8= datediff(DtDischAdjusted,DtLatRem,"days").
if TempLOS8 lt 8 ExLOS8=1.
end if.
execute.

* Age at entry or on first day is 18 or older.
compute ExAge18 = 0.
if AgemosyrsCat = 6 ExAge18 = 1.
execute.
* Age at entry or on first day is invalid (negative or > 21).
if AgemosyrsCat = 8888 ExAge18 = 8888.
execute.

variable labels
ExLOS8 'Episode has a LOS < 8 days'
ExAge18 'Child was age 18 or older at entry or on first day'.

value labels ExLOS8
0 'No'
1 'Yes'
```

```
8888 'Cannot be determined because LOS is negative or > 21'
/ ExAge18
0 'No'
1 'Yes'
8888 'Cannot be determined because age is negative or > 21'.
execute.

formats ExLOS8 ExAge18 (F1.0).


* Select the records to keep (i.e., 0 for both vars).
select if (ExLOS8 = 0 and ExAge18 = 0).
execute.
select if ExMissingDOD=0 and ExMissingPriorDOD=0.
EXECUTE.
*************************************************
* SUM EACH CHILD's LOS ACROSS EPISODES
*************************************************

sort cases BY ChildID.
aggregate
  /OUTFILE=* MODE=ADDVARIABLES
  /PRESORTED
  /BREAK=ChildID
  /Den_Child=SUM(LOSdays).

variable labels Den_Child 'Maltreatment in care denominator - Childs
total length of stay (days) across all episodes in the 12-month period'.
formats Den_Child (F3.0).
execute.

*flag children whose number of days in care for the year are greater than
365.

if den_child gt 365 over365 eq 1.
execute.
*************************************************

* These are due to data quality issues associated with some children who
were reported two or more times
* during the 12-month period, with different dates of latest removal, and
with no date of discharge in the first
* reported episode.


compute flagDCover = 0.
if Den_child gt 365 flagDCover = 1.
execute.

do if (Den_Child gt 366 AND (DtPeriodEnd=date.mdy(9,30,2012) or
DtPeriodEnd=date.mdy(3,31,2012) or DtPeriodEnd=date.mdy(9,30,2016) or
DtPeriodEnd=date.mdy(3,31,2016) or DtPeriodEnd=date.mdy(9,30,2020) or
DtPeriodEnd=date.mdy(3,31,2020))).
compute Den_Child=366.
```

```
end if.
EXECUTE.
do if (Den_Child gt 365 AND (DtPeriodEnd<>date.mdy(9,30,2012) and
DtPeriodEnd<>date.mdy(3,31,2012) and DtPeriodEnd<>date.mdy(9,30,2016) and
DtPeriodEnd<>date.mdy(3,31,2016) and DtPeriodEnd<>date.mdy(9,30,2020) and
DtPeriodEnd<>date.mdy(3,31,2020))).
compute Den_Child=365.
end if.
execute.



* Some children may have a total LOS = 0 days.
**************************************************

* These are children who exited on the first day of the 12-month period,
or entered on the last day of the
* 12-month period. At these points, they have not been in care during the
12-month period for a full 24 hours.
* In addition, victimization rates for these children cannot be
calculated when their LOS (i.e., denominator) is 0.

select if Den_Child <> 0.
execute.

**************************************************
* FINAL PREP OF AFCARS DATA
**************************************************

* Count the number of episodes. Maximum number should be 2 (one for each
6-month reporting period).
sort cases by ChildID DtLatRem.
compute numepisodes = 1.
if (ChildID = lag(ChildID))numepisodes = lag(numepisodes) + 1.
execute.

* This indicator excludes any maltreatment report that occurs within 7
days of removal from home (DtLatRem).
* Therefore, compute DtLatRemPlus7 to be the removal date plus seven
days. This will be used for later to
* exclude maltreatment reports in NCANDS that occured before
DtLatRemPlus7.


compute DtLatRemPlus7 =datesum(DtLatRem, 7, "days").
execute.
formats DtLatRemPlus7 (adate10).
variable labels DtLatRemPlus7 'Childs date of latest removal plus 7
days'.

* Later we need to find only maltreatment reports that occurred after the
date of latest removal and before
* the date of discharge. If the the child is still in care at the end of
the 12-month period, his DtDisch is missing,
```

```
* in which case we use DtPeriodEnd as the effective date of discharge.

if sysmis(DtDischAdjusted) DtDischAdjusted2=DtPeriodEnd.
if sysmis(DtDischAdjusted2) DtDischAdjusted2=DtDischAdjusted.
execute.
formats DtDischAdjusted2 (adate10).

variable labels DtDischAdjusted2 'Discharge date used for NCANDS
matching. For children who did not exit during the 12-month period,
DtPeriodEnd is their effective date of discharge'.

* Save.
save outfile= 'Temp_FH\Maltx FC 1 AFCARS duplicated ' + PeriodType +
YYstr + '.sav'
  /keep = state stateabb statetxt TwelveMoCohort recnumbr ChildID Ageyrs
Agemosyrs AgemosyrsCat sex DtLatRemPlus7 DtDischAdjusted2 Den_Child.
dataset name AFCARSDuplicated.

* Restructure AFCARS data from long to wide (one record per child)
*****************************************************

* Current AFCARS file has one record per episode during the 12-month
period, so a child with two episodes
* has two records. This new file will contain one record per child, with
all of his episode information on one
* record, indexed by N. For example: state ChildID recnumbr
DtDischAdjusted.1 DtDischAdjusted.2 ... etc.
* We will later match this file with NCANDS (matching on recnumbr, which
in NCANDS is called afcarsid).
* There will never be more than two episodes per child (i.e., variable.2)
that are indexed in the AFCARS wide file,
* since it is based on two 6-month periods and each period can have only
episode.

sort cases BY ChildID DtLatRemPlus7  .
casestovars
 /ID = ChildID
 /GROUPBY = VARIABLE.


* Flag records indicating they are from AFCARS
*****************************************************

* This will be used for denominator for determining the number of AFCARS
matches in the NCANDS file.
compute AFCARS = 1.
execute.

* Save.
sort cases by stateabb afcarsid.
save outfile ='Temp_FH\Maltx FC 2 AFCARS wide ' + PeriodType + YYstr +
'.sav'.

*****************************************************
```

```
*************************************************
* PREPARE NCANDS DATA
*************************************************
*************************************************
*Open Multiyear NCANDS file with derived variables.


get file 'NCANDS_Comb_FH\CFSR 3 NCANDS source data.zsav'
/keep subyr stateabb rptid chid rptdt chage chbdate afcarsid inciddt
flvictim statetxt state.
dataset name NCANDSChild window=front.
dataset close AFCARSDuplicated.

CACHE.


** Select relevant victim records for the FY of interest from two
consecutive Child Files.

select if ((subyr=YYYY or subyr = YYYY+1)) AND (RPTDT ge
date.mdy(10,1,YYYY-1) and rptdt le date.mdy(9,30,YYYY)) .
execute.

select if flvictim = 1.
execute.

sort cases by stateabb chid rptdt.
execute.
*************************************************
* Individual Level DQ - FLAG AND DROP STATES THAT EXCEED DQ LIMITS ON
NCANDS DQ CHECKS
*************************************************


get file 'NCANDS_DQ_FH\Merged NCANDS IND files.sav'.
dataset name DQResults_Ind window=front.

SELECT IF subyr = YYYY.
execute.

sort cases by stateabb chid.

Dataset activate NCANDSChild.

match files
    /FILE='NCANDSChild'
    /TABLE='DQResults_Ind'
    /BY stateabb chid.
execute.


SELECT IF sysmis(notsamechild) OR notsamechild eq 0.
EXECUTE.
```

```
************************************************
* SELECT APPLICABLE RECORDS
************************************************
* If child has a subsequent report, and it is within 1 day of his initial
report, we do not count it (i.e, reports are
* treated as 'rolled-up' into a single report).

sort cases by stateabb chid rptdt.
compute daysdiff2 = datediff(rptdt,lag(rptdt), "days").
EXECUTE.
select if (chid ne lag(chid) or daysdiff2 gt 1).
EXECUTE.


* Keep only records with an afcarsid.
select if AFCARSID ne ' '.
execute.


************************************************
* MERGE EPISODE DATA FROM AFCARS
INTO NCANDS TO IDENTIFY CHILDREN VICTIMIZED WHILE IN CARE
************************************************

* Merge fields from AFCARS wide file into NCANDS file. NCANDS file
includes one record for every report a
* child may have had; AFCARS file includes one record per child.
sort cases by stateabb afcarsid.
match files
   /file = *
   /table 'Temp_FH/Maltx FC 2 AFCARS wide ' + PeriodType + YYstr + '.sav'
   /by stateabb afcarsid.
execute.

* Identify children victimized while in care for any of his episodes (up
to two) in the 12-month period.
* At this point, the count of victimInCare will be duplicate since a
child has a record for every maltreatment report
* during the 12-month period. When we aggregate the file later and select
max(victimInCare), we will have a
* unique count of children who were a victim in care at any point during
the 12-month period.
compute victimInCare = 0.
if (rptdt ge DtLatRemPlus7.1 and rptdt lt DtDischAdjusted2.1)victimInCare
= 1.
if (rptdt ge DtLatRemPlus7.2 and rptdt lt DtDischAdjusted2.2)victimInCare
= 1.

* Count victimizations. For example, if a child has three reports in
NCANDS (and is a victim in each report), and
* all three reports were reported (or the incident occurred) during the
child's first episode (DtLatRemPlus7.1
```

```
* and DtDischAdjusted2.1), victimization1 will = 1 for all three
records/reports. When we aggregate the file later
* and sum the victimizations (e.g., sum(victimization1), the result will
show 3 victimizations for that one episode.
compute victimization1 = 0.
compute victimization2 = 0.
if (rptdt ge DtLatRemPlus7.1 and rptdt lt
DtDischAdjusted2.1)victimization1 =  1.
if (rptdt ge DtLatRemPlus7.2 and rptdt lt
DtDischAdjusted2.2)victimization2  = 1.
EXECUTE.

* Incident date adjustment. Recode victimInCare to 0 and victimization.N
to 0 if the incident date is present and
* shows the child was not victimized while in care (i.e., the actual date
of maltreatment was before date of latest
* removal plus 7, or after (or the same day of) the date of discharge.
do if ((victimization1=1) and not(missing(inciddt)) and ((inciddt lt
DtLatRemPlus7.1) or (inciddt ge DtDischAdjusted2.1))).
compute adjusted=1.
compute adjustvictim1=1.
compute victimization1=0.
compute victimIncare=0.
end if.
execute.
do if ((victimization2=1) and not(missing(inciddt)) and ((inciddt lt
DtLatRemPlus7.2) or (inciddt ge DtDischAdjusted2.2))).
compute adjusted=1.
compute adjustvictim2=1.
compute victimIncare=0.
compute victimization2=0.
end if.
execute.

* Restructure data from long to wide (one record per child)
**************************************************

* Current file has one record per maltreatment report during the 12-month
period, so a child with three
* reports has three records. This new file will contain one record per
child, with summary data from all of his
* maltreatment reports on one record. It will contain only victims in
care during the 12-month period; it will not
* contain children in care who were not victims. The file will hold
variables indicating if the NCANDS child has an
* AFCARSID in the NCANDS file (hasafcars = 1), also exists in the AFCARS
file (AFCARS = 1), was a victim
* in care at any point during the 12-month period (victimInCare = 1), and
how many victimizations he had for each of up
* to two possible foster care episodes (victimization1 and
victimization2). "AFCARS" is the only variables from
* the AFCARS file that is kept. Removed   /NCANDSvic 'Unique NCANDS
victim with AFCARS ID' = max(hasafcars)
```

```
dataset declare VictimsInCare.
aggregate outfile = 'VictimsInCare'
  /break = stateabb afcarsid
  /FFY 'Federal Fiscal Year' = first (subyr)
  /AFCARSMatch 'Unique NCANDS victims with AFCARS matches' = max(AFCARS)
  /victimInCare 'Unique count of children victimized while in care'  =
max(victimInCare)
  /victimization1 'victim report1' = sum(victimization1)
  /victimization2 'victim report2' = sum(victimization2).

dataset activate VictimsInCare.
dataset close NCANDSChild.

* Total up all victimizations for each child.

compute Num_Child = sum(victimization1 to victimization2).
execute.
variable labels Num_Child 'Number of victimizations child experienced
while in care across all episodes n the 12-month period'.

* Save.
save outfile = 'Temp_FH/Maltx FC 3 Unique victims in care ' + YYstr +
'.sav'.

************************************************
* MERGE VICTIMS IN CARE FROM NCANDS INTO AFCARS
************************************************

get file = 'Temp_FH/Maltx FC 2 AFCARS wide ' + PeriodType + YYstr +
'.sav'.
dataset name Unduplicated window=front.
dataset close VictimsInCare.

* Merge data for victims in care during the 12-month period into the
AFCARS file, which contains all children
* served in the 12-month period, but no information about their victim
status or presence/absense in NCANDS.
* This will merge in the following fields: FFY, NCANDSvic, AFCARSMatch,
victimInCare, victimization1,
* victimization2.
sort cases by stateabb afcarsid.
match files
   /file = *
   /file =  'Temp_FH/Maltx FC 3 Unique victims in care ' + YYstr + '.sav'
   /by stateabb afcarsid.
execute.

* File now includes children from NCANDS that were not in the AFCARS file
(presumably because they did not
* enter foster care and were never reported in AFCARS). These cases have
blanks for all their AFCARS variables.
* We want to get rid of these records; delete them.
select if (Den_Child ge 0).
execute.
```

```
* File now contains all children in AFCARS served during the 12-month
period. If the child was also in NCANDS,
* we have data about whether the child was a victim while in care during
the period (victimInCare) and the number
* of victimizations he experienced during his time in care (Num_Child).
If the child was not in NCANDS, these
* data are currently blank, and we assume - because the child was not
reported in NCANDS - that he is not a
* victim. Populate their data accordingly to show they were in AFCARS but
not in NCANDS (AFCARSMatch = 0),
* were not an NCANDS victim (NCANDSVic = 0), were not a victimincare
(victimInCare = 99), and had zero
* victimizations (Num_Child = 0).

do if missing(FFY).
compute FFY=YYYY.
compute NCANDSVic=0.
compute victimincare=9999.
compute Num_Child=0.
end if.
execute.
missing values victimInCare (9999).


*************************************************
* SUMMARY STATISTICS
*************************************************

* Create variables holding state and national performance.
*************************************************

* Child-level data.
compute Perf_Child = (Num_Child / Den_Child).
compute Perf_Child_MP = (Num_Child / Den_Child) * 365.
execute.

* State-level data.
sort cases by state (A) ChildID (A).
aggregate
  /OUTFILE=* MODE=ADDVARIABLES
  /PRESORTED
  /BREAK=state
  /Num_State=SUM(Num_Child)
  /Den_State=SUM(Den_Child)
  /N_State=N.
compute Perf_State = (Num_State / Den_State).
compute Perf_State_MP = (Num_State / Den_State) * 100000.
execute.

* National-level data.
aggregate
  /OUTFILE=* MODE=ADDVARIABLES
  /BREAK=
```

```
  /Num_Nation=SUM(Num_Child)
  /Den_Nation=SUM(Den_Child)
  /N_Nation=N.
compute Perf_Nation = (Num_Nation / Den_Nation).
compute Perf_Nation_MP = (Num_Nation / Den_Nation) * 100000.
execute.

formats Perf_Child Perf_State Perf_Nation (F6.5).

variable labels
Perf_Child 'Maltx in care - Victimization rate per days in care (child)'
Perf_Child_MP 'Maltx in care - Victimization rate per 365 days in care
(child)'
N_State 'Number of children served in the state during the 12-month
period'
Den_State 'Maltx in care denominator - Total number days children were in
care in state across all episides in the 12-month period'
Num_State 'Maltx in care numerator - Total number of victimizations in
state children experienced while in care across all episodes in the 12-
month period'
Perf_State_MP 'Maltx in care - Victimization rate per 100,000 days in
care (state)'
N_Nation 'Number of children served in the nation during the 12-month
period'
Den_Nation 'Maltx in care denominator - Total number days children were
in care in nation across all episides in the 12-month period'
Num_Nation 'Maltx in care numerator - Total number of victimizations in
nation children experienced while in care across all episodes in the 12-
month period'
Perf_Nation_MP 'Maltx in care - Victimization rate per 100,000 days in
care (nation)'.


************************************************
* OUTPUT FILES
************************************************

* Save.
save outfile='Output_Child_FH\CFSR 3 - Observed perf for maltx in care '
+ PeriodType + YYstr + '.sav'
   /compressed.

* Save file for STATA (for multi-level modeling).
* If child has two episodes during the period, use age associated with
the first episode.
rename variables (Agemosyrs.1 = ChildAge).
save translate OUTFILE='Output_Child_FH\CFSR 3 - Observed perf for maltx
in care ' + PeriodType + YYstr + '.dta'
  /TYPE=STATA
  /VERSION=8
  /EDITION=SE
  /MAP
  /REPLACE
```

```
  /KEEP=state stateabb statetxt TwelveMoCohort ChildID Num_Child
Den_Child ChildAge N_State Den_State
Num_State Perf_State Perf_State_MP N_Nation Den_Nation Num_Nation
Perf_Nation Perf_Nation_MP.
rename variables (ChildAge = Agemosyrs.1).

* Create file with one record per state holding observed performance for
the 12-month cohort.
compute numstates = 1.
if (state eq lag(state))numstates = 0.
execute.

dataset copy OneRecordPerState.
dataset activate OneRecordPerState.
select if (numstates=1).
execute.
dataset activate OneRecordPerState.

string Indicator (A30).
execute.
compute Indicator = "Maltx in care".
execute.

save outfile='Output_State_FH\CFSR 3 - Observed perf for maltx in care
State file ' + PeriodType + YYstr + '.sav'
   /keep state stateabb statetxt Indicator TwelveMoCohort Perf_State.

dataset activate Unduplicated.
dataset close OneRecordPerState.

* Save output.
output save OUTFILE='Output_State_FH\CFSR 3 - Observed perf for maltx in
care ' + PeriodType + YYstr + '.spv'.

************************************************************
ALL DONE.
************************************************************
```

CFSR 3: OBSERVED PERFORMANCE FOR RECURRENCE OF MALTREATMENT

```
* Encoding: UTF-8.
************************************************
CFSR 3: OBSERVED PERFORMANCE
- RECURRENCE OF MALTREATMENT.
************************************************.

set printback=on.

*The file handle Root_FH represents the root folder for the CFSR data.
file handle Root_FH /name="C:\CFSR 3\".
*The file handle CombinedFiles_FH is the input folder for this code. It
should contain the combined NCANDS and combined AFCARS source files
produced by earlier code.
file handle NCANDS_Comb_FH  /name="Root_FH\0 - Source Data".
* The file handle DQfiles_FH is the output folder for the results of the
DQ checks.
file handle NCANDS_DQ_FH /name= "Root_FH\2 - DQ NCANDS".
* The file handle SpssLogs_FH  is the output folder for the SPSS logs.
file handle SpssLogs_FH /name="Root_FH\5 - SPSS Logs".
* The file handle Observed_Child_FH is the output folder for the primary
results of this code (results on a per child basis).
file handle Observed_Child_FH /name="Root_FH\3 - Performance observed
child".
* The file handle Observed_State_FH is the output folder for the results
of this code (aggregated to a state-level basis).
file handle Observed_State_FH /name="Root_FH\3 - Performance observed
state".


*********************************************************************
******************************
SPECIFY THE 12-MONTH COHORT WHOSE OUTCOME WILL BE ASSESSED
*********************************************************************
******************************
     ********  START USER INPUT ********.
define YYYY () XXXX
!enddefine.
define YYstr () "XX"
!enddefine.
define YY2str () "XX"
!enddefine.

     ********  END USER INPUT ********

*********************************************************************
******************************
GET SOURCE FILES
*********************************************************************
******************************

GET file='NCANDS_Comb_FH\CFSR 3 NCANDS source data.zsav'
    /keep= subyr stateabb state statetxt rptid rptdt rptcnty inciddt chid
chage maldeath chbdate chsex afcarsid flvictim.
CACHE.
```

```
DATASET NAME Indicator.

************************************************
* FILE PREP
************************************************

* Verify the records represent the FYs of interest.
select if subyr eq YYYY OR subyr eq YYYY+1.
execute.

* Select only records with report dates on or after the beginning of the
first fiscal year.
select if rptdt ge date.mdy (10,01,YYYY-1).
execute.

 * Select down to only records with a victim flag = 1. (Only use
victimized children).
Select if flvictim = 1.
execute.

* Create variable that indicates the user-specified 12-month cohort.
string TwelveMoCohort (A6).
compute TwelveMoCohort = concat("FY",String(YYYY,F4)).
execute.

*select children only ago 0 thru 17.
select if chage ge 0 AND chage le 17.
EXECUTE.

************************************************************************
******************************
* State Level DQ - FLAG AND DROP STATES THAT EXCEED DQ LIMITS ON DQ
CHECKS
************************************************************************
******************************

get file 'NCANDS_DQ_FH\Merged NCANDS DQ files.sav'.
dataset name DQResultsN window=front.

* Prevent accidentally overwriting file.
dataset copy  DQIndicatorN.
dataset activate DQIndicatorN.
dataset close DQResultsN.

*Create a flag variable for records applicable for the current run.
if ((malREC eq 1) and (DQFileXW eq 'Cross file') and (subyr = YYYY))
DQchecksApplyN =1.
if ((malREC eq 1) and (DQFileXW eq 'Within file') and (subyr ge YYYY) AND
(subyr le YYYY+1)) DQchecksApplyN =1.
select if DQchecksApplyN =1.
sort cases by stateabb subyr.
execute.
```

```
*Create a new table with state abbreviation and a flag variable for each
state denoting the number of relevant DQ checks that were failed.
DATASET DECLARE DQ_State_Exclusion.
aggregate
  /OUTFILE=DQ_State_Exclusion
  /BREAK=stateabb
  /DQFailedCheckN_sum=SUM(DQFailedCheck).

*  Merge the number of DQ checks failed into the original indicator data
file.
dataset activate indicator.
sort cases by stateabb.
match files
    /FILE=*
    /TABLE='DQ_State_Exclusion'
    /BY stateabb.
execute.

*Close datasets that are no longer needed and exclude any state that
failed at least one DQ check.
dataset close DQIndicatorN.
dataset close DQ_State_Exclusion.

select if DQFailedCheckN_sum eq 0.
execute.
delete variables DQFailedCheckN_sum.

*************************************************
* Individual Level DQ - FLAG AND DROP STATES THAT EXCEED DQ LIMITS ON DQ
CHECKS
*************************************************


get file 'NCANDS_DQ_FH\Merged NCANDS IND files.sav'.
dataset name DQResults_Ind window=front.

SELECT IF subyr = YYYY.
execute.

sort cases by stateabb chid.

Dataset activate indicator.

sort cases by stateabb chid rptdt.

match files
    /FILE=*
    /TABLE='DQResults_Ind'
    /BY stateabb chid.
execute.

Dataset close DQResults_Ind.

SELECT IF sysmis(notsamechild) or notsamechild=0.
```

```
EXECUTE.

************************************************
* DERIVED VARIABLES AND PROCESSING
************************************************

* Determine the time between reports for 14 day rollup.
do if (stateabb = lag(stateabb) and chid = lag (chid)).
compute timebetweenrpts = datediff ( rptdt,lag(rptdt), 'days').
end if.

* Apply 14 day roll up.
recode timebetweenrpts (sysmis = 9999).
select if timebetweenrpts gt 14.

* Count the relative report position for a victim.
compute posit = 1.
if (stateabb eq lag(stateabb) and chid eq lag(chid))posit = lag(posit)+1.
execute.

* Exclude fatalities if they happened in the first report. * Possible
issue with maltreatment death based of Federal Register explanation.
compute keep = 1.
if (posit eq 1 and maldeath eq 1) keep = 0.
execute.
select if keep = 1.
execute.

*Exclude records with matching incident dates.
compute sameincd =0.
do if (stateabb = lag(stateabb) and chid = lag (chid)).
if (not(missing(inciddt)) AND not(missing(lag(inciddt))) AND (inciddt eq
lag(inciddt))) sameincd =1.
end if.
execute.
select if sameincd ne 1.
execute.


compute posit2 = 1.
if (stateabb eq lag(stateabb) and chid eq lag(chid))posit2 =
lag(posit2)+1.
execute.
* Select only the first two reports for a victim. Recurrence is only
computed using the first two reports, so
* subsequent reports are of no interest.
select if (posit2 le 2).
execute.

compute missingincd = 0.
if missing(inciddt) missingincd = 1.
execute.

* Aggregate to create a unique file.
```

```
dataset declare SourceData_Aggregated.
aggregate outfile = 'SourceData_Aggregated'
  /break = stateabb state chid
  /TwelveMoCohort = first(TwelveMoCohort)
  /FY = first(subyr)
  /county = first(rptcnty)
  /dob = first (chbdate)
  /ageinyears = first (chage)
  /initialrpt = first(rptdt)
  /lastrpt = last(rptdt)
  /initialincdt = first(inciddt)
  /lastincdt = last(inciddt)
  /missingincddt = max(missingincd)
  /statetxt = first(statetxt).
dataset activate SourceData_Aggregated.
dataset close Indicator.

 if (missingincddt eq 1) initialincdt = initialrpt.
 if (missingincddt eq 1) lastincdt = lastrpt.
 execute.

************************************************
* Flag cases where recurrence occurs.
************************************************

* Compute the time between the first and the last report.
compute mthsbetweenrpts = datediff(lastrpt,initialrpt, 'months')..
if (initialrpt eq lastrpt) mthsbetweenrpts = $sysmis.

* If the time betweeen reports is greater than or equal to 0 (we have
already applied 14 day roll up rule and removed reports) and less than 12
months, recurrence occurs.
compute recur = 0.
if (initialrpt <> lastrpt) AND (mthsbetweenrpts ge 0 and mthsbetweenrpts
lt 12) recur = 1.
execute.

* If the initial incident date is the same as the last incident date, set
recur to 0.
do if (initialincdt ge 1 and lastincdt ge 1).
if (initialincdt eq lastincdt)recur = 0.
end if.
execute.

variable labels recur 'Maltx recurrence numerator - Child had a
recurrence of maltx within 12 months of his initial report'.
value labels recur
   0 'No recurrence occurred'
   1 'Yes recurrence occurred'.
formats recur (F1.0).

* Select only records where the first report date occurred in the first
fiscal year (this uses two submission periods).
select if (initialrpt le date.mdy(09,30,YYYY)).
```

```
execute.

*rename variables (chid ageinmonths ageinyears recur = ChildID AgeMos
AgeYrs Num_Child).
rename variables (chid recur = ChildID Num_Child).

************************************************
* CALCULATE AGE AT INITIAL REPORT
************************************************

* Compute age in months at the time of the initial report.
compute ageinmonths = datediff(initialrpt,dob, 'months').

* Some children have a negative ageinmonths due to dob > initialrpt.
These are plausible and involve allegations
* concerning children in utero. Recode to 0 for 0-3 months age group,
which includes unborn.
recode ageinmonths (lowest thru -1 = 0).
variable labels ageinmonths 'Age in months at initial report'.
value labels ageinmonths 0 'less than 1 month includes unborn'.
variable labels ageinyears 'Age in years at initial report'.
value labels ageinyears 0 'Less than1 year includes unborn' 99 'Unknown
age'.

* Recode age in months to desired categories.
* Age at initial victimization (interval), specified as 0-3 mos, 4-11
mos, 1 yr ... 21 yrs.
* Used later in risk adjustment, where each age value is converted to a
yes/no (1/0) dummy variable.
recode ageinmonths (216 thru 263=6) (204 thru 216=5) (132 thru 204=4) (72
thru 132=3) (12 thru 72=2) (4 thru 12=1) (0 thru 4=0) (else=sysmis) into
AgeMosyrsCat.

* Children < 1 currently have a value of 0 for AgeYrs. Need to split
these into 0-3 mos (0) mos and 4-11 mos (1).
* To make room for the 0 and 1 for < 1 children, make a copy of AgeYrs
then shift all subsequent ages, so
* age1 = 2, age2 = 3, etc.
compute AgeMosyrs = ageinyears.
* Recode 1 - 21 to 2 - 22.
if (ageinyears ge 1) AgeMosyrs = ageinyears + 1.
if AgeMosyrsCat = 0 AgeMosyrs = 0.
if AgeMosyrsCat = 1 AgeMosyrs = 1.
execute.

variable labels
ageinmonths 'Age at initial report in months'
ageinyears 'Age at initial report in years (Under 1, 1, 2, 3 ... 21)'
AgeMosyrs 'Age at initial report in months (0-3 mo, 4-11 mo) and years
(1, 2, 3 ... 21)'
AgeMosyrsCat 'Age at initial report in months (0-3 mo, 4-11 mo) and years
in categories (1-5, 6-10, etc...)'.

* Create macro that holds age values.
```

```
DEFINE !AgeYrs ( )
0 '< 1 yr'
1 '1 yr'
2 '2 yrs'
3 '3 yrs'
4 '4 yrs'
5 '5 yrs'
6 '6 yrs'
7 '7 yrs'
8 '8 yrs'
9 '9 yrs'
10 '10 yrs'
11 '11 yrs'
12 '12 yrs'
13 '13 yrs'
14 '14 yrs'
15 '15 yrs'
16 '16 yrs'
17 '17 yrs'
18 '18 yrs'
19 '19 yrs'
20 '20 yrs'
21 '21 yrs'
!ENDDEFINE.

DEFINE !AgeMosYrs ( )
0 '0-3 mos'
1 '4-11 mos'
2 '1 yr'
3 '2 yrs'
4 '3 yrs'
5 '4 yrs'
6 '5 yrs'
7 '6 yrs'
8 '7 yrs'
9 '8 yrs'
10 '9 yrs'
11 '10 yrs'
12 '11 yrs'
13 '12 yrs'
14 '13 yrs'
15 '14 yrs'
16 '15 yrs'
17 '16 yrs'
18 '17 yrs'
19 '18 yrs'
20 '19 yrs'
21 '20 yrs'
22 '21 yrs'
!ENDDEFINE.

DEFINE !AgeMosYrsCat ( )
0 '0-3 mos'
1 '4-11 mos'
```

```
2 '1-5 yrs'
3 '6-10 yrs'
4 '11-16 yrs'
5 '17'
6 '18-21 yrs'
!ENDDEFINE.

value labels
ageinmonths
8888 'Invalid (age is negative or > 21)'
9999 'Cannot calculate (DOB missing)'
/ageinyears
!AgeYrs
8888 'Invalid (age is negative or > 21)'
9999 'Cannot calculate (DOB missing)'
/AgeMosyrs
!AgeMosYrs
8888 'Invalid (age is negative or > 21)'
9999 'Cannot calculate (DOB missing)'
/AgeMosyrsCat
!AgeMosYrsCat
8888 'Invalid (age is negative or > 21)'
9999 'Cannot calculate (DOB missing)'.

formats ageinyears ageinmonths AgeMosyrsCat AgeMosyrs (F3.0).

*****************************************************
* Create additional variables needed for Stata or Tableau.
*****************************************************

String Period (A10).
Compute Period = concate("FY",YYstr,"-FY",YY2str).
Execute.

* State-level data.
sort cases by state (A) ChildID (A).
aggregate
  /OUTFILE=* MODE=ADDVARIABLES
  /PRESORTED
  /BREAK=state
  /Num_State=SUM(Num_Child)
  /Den_State=N.
compute Perf_State = Num_State / Den_State.
compute Perf_State_MP = (Num_State / Den_State) * 100.
execute.

* National-level data.
aggregate
  /OUTFILE=* MODE=ADDVARIABLES
  /BREAK=
  /Num_Nation=SUM(Num_Child)
  /Den_Nation=N.
compute Perf_Nation = Num_Nation / Den_Nation.
compute Perf_Nation_MP =  (Num_Nation / Den_Nation) * 100.
```

```
execute.
formats Perf_State Perf_Nation (F8.5).

variable labels
Den_State 'Maltx recurrence denominator - Number of children in state who
were victimized in the first 12-month period'
Num_State 'Maltx recurrence numerator - Among children in the
denominator, number of children in state who had a recurerence of maltx
within 12 months of their initial victimization'
Perf_State 'Maltx recurrence - Percentage of children in state who had a
recurrence of maltx within 12 months of their initial victimization'
Den_Nation 'Maltx recurrence denominator - Number of children in nation
who were victimized in the first 12-month period'
Num_Nation 'Maltx recurrence numerator - Among children in the
denominator, number of children in nation who had a recurerence of maltx
within 12 months of their initial victimization'
Perf_Nation 'Maltx recurrence - Percentage of children in nation who had
a recurrence of maltx within 12 months of their initial victimization'.

*************************************************
* OUTPUT FILES
*************************************************

* Save.
save outfile='Observed_Child_FH\CFSR 3 - Observed perf for maltx
recurrence '  + YYstr + YY2str + '.sav'
   /compressed.

* Save file for STATA (for multi-level modeling).
save translate OUTFILE='Observed_Child_FH\CFSR 3 - Observed perf for
maltx recurrence ' + YYstr + YY2str + '.dta'
  /TYPE=STATA
  /VERSION=13
  /EDITION=SE
  /MAP
  /REPLACE
  /RENAME (AgeMosyrs = ChildAge)
  /KEEP=state stateabb statetxt TwelveMoCohort ChildID Num_Child ChildAge
Den_State
Num_State Perf_State Den_Nation Num_Nation Perf_Nation.

* Create file with one record per state holding observed performance for
the 12-month cohort.
compute numstates = 1.
if (state eq lag(state))numstates = 0.
execute.

dataset copy OneRecordPerState.
dataset activate OneRecordPerState.
select if (numstates=1).
execute.
dataset activate OneRecordPerState.

string Indicator (A30).
```

```
execute.
compute Indicator = "Maltx recurrence".
execute.

save outfile='Observed_State_FH\CFSR 3 - Observed perf for maltx
recurrence State file ' + YYstr + YY2str + '.sav'
   /keep state stateabb statetxt Indicator TwelveMoCohort Perf_State.

dataset activate SourceData_Aggregated.
dataset close OneRecordPerState.

output save OUTFILE='SpssLogs_FH\CFSR 3 - Observed perf for maltx
recurrence ' + YYstr + YY2str + '.spv'.


**************************************************
ALL DONE.
**************************************************
```

CFSR 3: OBSERVED PERFORMANCE - PERMANENCY IN 12 MONTHS FOR CHILDREN ENTERING FOSTER CARE and RE-ENTRY TO FOSTER CARE IN 12 MONTHS

```
* Encoding: UTF-8.
* Encoding: .

set printback=on.
*The file handle Root_FH represents the root folder for the CFSR
data.***** CHANGE THE ROOT FILE HANDLE TO MATCH THE WHERE YOU HAVE PLACED
THE CFSR 3 FOLDER ON YOUR COMPUTER***.
file handle Root_FH /name="C:\CFSR 3".
file handle AFCARS_Comb_FH  /name="Root_FH\0 - Source Data".
file handle AFCARS_DQ_FH /name= "Root_FH\2 - DQ AFCARS".
file handle SpssLogs_FH /name="Root_FH\5 - SPSS Logs".
file handle Observed_Child_FH /name="Root_FH\3 - Performance observed
child".
file handle Observed_State_FH /name="Root_FH\3 - Performance observed
state".
file handle Fixed_FH /name ="Root_FH\1 - Fixed Files".

************************************************
CFSR 3: OBSERVED PERFORMANCE
- PERMANENCY IN 12 MONTHS FOR CHILDREN ENTERING FOSTER CARE
- RE-ENTRY TO FOSTER CARE IN 12 MONTHS.
************************************************

* Identify the 12-month cohort of interest (e.g., children entering in
11B12A)
* by entering "AB" or "BA" for PeriodType and the year the 12-month
period ends (for YYYY and YYstr).

* AB period (A file + B file) spans Oct 1 - Sept 30 of the following
year.
* BA period (B file + A file) spans Apr 1 - Mar 31 of the following year.

* Examples:
* ( * ) represents a FFY
* Children who entered     PeriodType  YYYY   YYstr
* 09B10A                        BA              2010     10
* 10A10B *                      AB              2010     10
* 10B11A                        BA              2011     11
* 11A11B *                      AB              2011     11
* 11B12A                        BA              2012     12
* 12A12B *                      AB              2012     12
* 12B13A                        BA              2013     13
* 13A13B *                      AB              2013     13
* 13B14A                        BA              2014     14
* etc.

********  START USER INPUT ********

define PeriodType () "XX"
!enddefine.
define YYYY () XXXX.
!enddefine.
define YYstr () "XX"
!enddefine.
```

```
********  END USER INPUT ********

***************************************************
GET SOURCE FILE
***************************************************

* Open source data.
get file 'AFCARS_Comb_FH/CFSR 3 AFCARS source data.sav'.
dataset name Indicator window=front.

* Delete unused variables.
delete variables amiakn asian blkafram hawaiipi white untodetm hisorgin
clindis mr vishear phydis dsmIII othermed everadpt ageadopt manrem
phyabuse sexabuse neglect aaparent
daparent aachild dachild childis chbehprb prtsdied prtsjail nocope
abandmnt relinqsh housing placeout casegoal ctkfamst ctk1yr ctk2yr
fosfamst fcctk1yr fcctk2yr rf1amakn
rf1asian rf1blkaa rf1nhopi rf1white rf1utod hofcctk1 rf2amakn rf2asian
rf2blkaa rf2nhopi rf2white rf2utod hofcctk2 ivefc iveaa ivaafdc ivdchsup
xixmedcd ssiother noa fcmntpay DtReview
DtLatRemTrans DtTPRMom DtTPRDad DtDischTrans.

***************************************************
CREATE DATE PARAMETERS (In AFCARS Source File)
***************************************************

* Start date of a 12-month period (10/1/YYYY or 4/1/YYYY).
* End date of a 12-month period (3/31/YYYY or 9/30/YYYY).

* Perm in 12 (Entries) involves following children who entered during the
12-month period specified earlier.
* Identify the start date (DtPeriodBeg) and end date (DtPeriodEnd) that
defines this 12-month period.

* Perm in 12 (Entries) and Re-entry indicators require data spanning
three years, beginning with the 12-month
* period specified earlier. Identify the start date (DtPeriodBeg) and end
date (DtMeasureEnd) that defines these
* three years. Also, specify the end date for the 6-month period prior to
DtMeasureEnd
* (DtMeasureEnd_Minus6Mo). This will be used for the cross-file DQ checks
which apply to all but the last
* 6-month period.

do if PeriodType = "AB".
compute DtPeriodBeg=date.mdy(10,01,YYYY-1).
compute DtPeriodEnd=date.mdy(09,30,YYYY).
compute DtMeasureEnd=date.mdy(09,30,YYYY+2).
compute DtMeasureEnd_Minus6Mo=date.mdy(03,31,YYYY+2).
end if.
do if PeriodType = "BA".
compute DtPeriodBeg=date.mdy(04,01,YYYY-1).
compute DtPeriodEnd=date.mdy(03,31,YYYY).
```

```
compute DtMeasureEnd=date.mdy(03,31,YYYY+2).
compute DtMeasureEnd_Minus6Mo=date.mdy(09,30,YYYY+1).
end if.
execute.
formats DtPeriodBeg DtPeriodEnd DtMeasureEnd DtMeasureEnd_Minus6Mo
(adate10).

variable labels
DtPeriodBeg 'Start date of the 12-month period specified'
DtPeriodEnd 'End date of the 12-month period specified'
DtMeasureEnd 'End date of the period needed to observe performance'
DtMeasureEnd_Minus6Mo 'End date of the period needed to observe
performance, minus 6 months'.

string TwelveMoCohort (A4).
compute TwelveMoCohort = concat(PeriodType,YYstr).
execute.

*************************************************
* Remove records outside the report period
*************************************************

if (DtReportBeg ge DtPeriodBeg) and (DtReportEnd le DtMeasureEnd)
ReportedDuringPeriod = 1.
execute.

* Select the records to keep.
select if ReportedDuringPeriod=1.
execute.

delete variables ReportedDuringPeriod.


*************************************************
REMOVE STATES THAT EXCEED DQ LIMITS
*************************************************

* The DQ checks are applied only to the 6-month periods that fall between
DtPeriodBeg and DtMeasureEnd,
* except the cross file checks, which are not applied to the last 6-month
period.

* Open the data quality results done previously.
get file 'AFCARS_DQ_FH\Merged AFCARS DQ files.sav'.
dataset name DQResults window=front.

* Prevent accidentally overwriting file.
dataset copy  DQIndicator.
dataset activate DQIndicator.
dataset close DQResults.

* Create variable that indicates the user-specified 12-month cohort
(e.g., "BA12" represents "11B12A").
string TwelveMoCohort (A4).
```

```
compute TwelveMoCohort = concat(PeriodType,YYstr).
execute.

*************************************************
CREATE DATE PARAMETERS (In The DQ File)
*************************************************

* Start date of a 12-month period (10/1/YYYY or 4/1/YYYY).
* End date of a 12-month period (3/31/YYYY or 9/30/YYYY).

* Perm in 12 (Entries) involves following children who entered during the
12-month period specified earlier.
* Identify the start date (DtPeriodBeg) and end date (DtPeriodEnd) that
defines this 12-month period.

* Perm in 12 (Entries) and Re-entry indicators require data spanning
three years, beginning with the 12-month
* period specified earlier. Identify the start date (DtPeriodBeg) and end
date (DtMeasureEnd) that defines these
* three years. Also, specify the end date for the 6-month period prior to
DtMeasureEnd
* (DtMeasureEnd_Minus6Mo). This will be used for the cross-file DQ checks
which apply to all but the last
* 6-month period.

do if PeriodType = "AB".
compute DtPeriodBeg=date.mdy(10,01,YYYY-1).
compute DtPeriodEnd=date.mdy(09,30,YYYY).
compute DtMeasureEnd=date.mdy(09,30,YYYY+2).
compute DtMeasureEnd_Minus6Mo=date.mdy(03,31,YYYY+2).
end if.
do if PeriodType = "BA".
compute DtPeriodBeg=date.mdy(04,01,YYYY-1).
compute DtPeriodEnd=date.mdy(03,31,YYYY).
compute DtMeasureEnd=date.mdy(03,31,YYYY+2).
compute DtMeasureEnd_Minus6Mo=date.mdy(09,30,YYYY+1).
end if.
execute.
formats DtPeriodBeg DtPeriodEnd DtMeasureEnd DtMeasureEnd_Minus6Mo
(adate10).

variable labels
DtPeriodBeg 'Start date of the 12-month period specified'
DtPeriodEnd 'End date of the 12-month period specified'
DtMeasureEnd 'End date of the period needed to observe performance'
DtMeasureEnd_Minus6Mo 'End date of the period needed to observe
performance, minus 6 months'.

*************************************************
* Select the AFCARS checks for this indicator and DQ results for the 6-
month period(s).
*************************************************

* ... Cross-file checks (apply to all but the last 6-month period).
```

```
if PermReEntry = 1 and DQFileXW = "Cross file" AND (DtReportBeg ge
DtPeriodBeg) and (DtReportEnd le DtMeasureEnd_Minus6Mo) DQChecksApply =
1.
* ... Within-file checks (apply to all 6-month periods).
if PermReEntry = 1 and  DQFileXW = "Within file" AND (DtReportBeg ge
DtPeriodBeg) and (DtReportEnd le DtMeasureEnd) DQChecksApply = 1.
execute.

* For each state, for the checks that apply, sum the number of checks it
failed.
sort cases by state.
select if DQChecksApply = 1.
execute.

DATASET DECLARE DQStateInd.
aggregate
  /OUTFILE=DQStateInd
  /BREAK=state
  /DQFailedCheck_sum=SUM(DQFailedCheck).
dataset activate DQStateInd.
dataset close DQIndicator.
* If DQFailedCheck_sum > 0, then state failed at least one of the checks
for at least one of the 6-month periods.
* List those states, the checks they failed, the % of problem cases, and
the periods in which the checks failed.
* Merge states' DQFailedCheck_sum value into Indicator file.
* Any state where DQFailedCheck_sum > 0 will be dropped from the dataset
and excluded from analyses.

*************************************************
* Merge the DQ results into the AFCARS Source File and drop all records
for states that fail the state-thresholds.
*************************************************

* ... Merge in all variables from DQIndicator file, then keep only
DQFailedCheck_sum.
dataset activate Indicator.
sort cases by state.

match files
    /FILE=*
    /TABLE='DQStateInd'
    /BY state
    /KEEP state to TwelveMoCohort DQFailedCheck_sum.
execute.

dataset close DQStateInd.

* Select only states that met the DQ checks.
select if DQFailedCheck_sum = 0.
execute.
delete variables DQFailedCheck_sum.
```

```
*********************************************************************
*****
* Further select down to children who entered in the first 12-month
period
*********************************************************************
*****
* This is the cohort of children for whom performance will be examined.

* Flag records with entry in the inital 12 month period.
COMPUTE Entered = 0.
if ((DtLatRem ge DtPeriodBeg) and (DtLatRem le DtPeriodEnd)) Entered = 1.
execute.

variable labels Entered 'Child entered care during the specified 12-month
period'.
value labels Entered
0 'No'
1 'Yes'.
EXECUTE.

*Flag all records for a kid with an entry in the initial 12 month period.
aggregate
/OUTFILE=* MODE=ADDVARIABLES
/break childid
/hasentered = max(entered).

*Drop all non-applicable records.
SELECT IF hasentered eq 1.
execute.

*********************************************************************
*****
* Select only the most recent 6-month record reported for each child, for
each episode.
*********************************************************************
*****

* This will create an episode-level file. Episodes are distinguished by
the date of latest removal from home.
* A child has a record in each 6-month submission until he discharged,
dropped, or was still in care as of
* the last reporting period we have in the file. We only want one record
per child, per episode, and the record we
* pick is the last one reported for that episode. This record will
contain the most recent data available that
* describes the child's episode (e.g., LOS, age at entry, etc.).

* Flag the records to keep.
sort cases BY ChildID (A) DtLatRem (A).
match files
  /FILE=*
  /BY ChildID DtLatRem
  /LAST=Flag_LastRpt4Ep.
execute.
```

```
* Select the last 6-month record for each child, for each episode.
select if Flag_LastRpt4Ep=1.
execute.

delete variables Flag_LastRpt4Ep.

*************************************************************************
*****
*     If a child has two episodes, but those records have dates that
indicate the episodes overlap, only the most recent record is used
*************************************************************************
*****

* Identify the number of episodes for each child.
sort cases by ChildID dtreportbeg.
if (childid eq lag(childid)) duplicatechild=1.
execute.
compute position=1.
if (duplicatechild=1) position = lag(position)+1.
execute.

if missing(duplicatechild) duplicatechild eq 0.
execute.
aggregate
outfile=* mode=ADDVARIABLES
/break state childid
/dupchild = max(duplicatechild).

* Flag and drop the first episode when two episodes exist, and the first
one failed to discharge before a new episode started (and consequentially
they operlap).
sort cases by ChildID dtreportbeg.
if (dtlatrem ne lag(dtlatrem) AND childID eq lag(ChildID) and
(missing(lag(dtdischadjusted)) OR lag(Adjust18) eq 1)) flagCase eq 1.
execute.

if flagcase eq 1 AND (missing(dtpriordisch) or (dtpriordisch lt
dtreportbeg)) flagcase2 eq 1.
execute.

aggregate
  /OUTFILE=* MODE=ADDVARIABLES
  /BREAK=ChildID
  /Flagrptremove2 = max(flagcase2).

compute dropreport eq 0.
if (flagrptremove2 eq 1 AND position eq 1) dropreport eq 1.
execute.

select if dropreport eq 0.
execute.
```

```
*Flag and drop the first episode when two episodes exist, and the second
epsidoes date of latest removal occurs before the first episodes data of
discharge (and consequentially they overlap).
if childid eq lag(childid) AND dtlatrem lt lag(dtdisch) AND
not(missing(lag(dtdisch))) overlapep eq 1.
execute.

aggregate
  /OUTFILE=* MODE=ADDVARIABLES
  /BREAK=ChildID
  /overlapepchild = max(overlapep).

compute dropoverlap eq 0.
if (overlapepchild eq 1 AND position eq 1) dropoverlap = 1.
execute.

select if dropoverlap eq 0.
execute.

************************************************
* A. Calculate time to reentry.
************************************************
* ... Before we select only children who entered during the 12-month
period specified earlier, we need to
* calculate time to reentry, which needs to consider subsequent entries
that occurred in the following
* year or years. For children with more than 1 removal, time to reentry
is the time in months between
* the DtPriorDisch and DtLatRem for a given episode.

do if totalrem > 1 AND ageNyrs lt 18 AND position >1.
if DtLatRem>DtPriorDisch
TimeToReentry_temp=datediff(DtLatRem,DtPriorDisch,"months").
end if.
execute.

*create flag that will be used to identify children who dischage and
reenter in same 6mo period.
if (dtpriordisch ge DtReportBeg AND dtpriordisch le DtReportEnd)
priorsameperiod =1.
execute.

* ... Copy the value of TimeToReentry_temp to the child's previous
episode record. We do this because it's
* the time from the *previous* episode's exit to the next entry we are
interested in. That previous episode will
* have the child's age at entry, LOS, etc.; all it needs now is the time
to reentry, if one exists.
* In some cases, the first record we have for a child has a totalrem > 1.
This is usually because the state has
* not reported data for the child's earlier episodes. For these records,
there is no previous entry to assign the
* time to, so we set TimeToReentry_temp to missing. If we don't, the lead
command will put it in the record for
```

```
* the previous child. The file should already be sorted by ChildID,
DtLatRem.
PRESERVE.
set workspace 200000.
if (ChildID<>lag(ChildID)) TimeToReentry_temp = $sysmis.
create TimetoNextReentry=Lead(TimeToReentry_temp,1).
if (ChildID<>lag(ChildID)) priorsameperiod = $sysmis.
create FlagforDeletion=Lead(priorsameperiod,1).
execute.
RESTORE.

delete variables TimeToReentry_temp.

*Flag and drop cases that discharge and re-enter in the same six month
period. We will not have necessary discharge reason information when this
happens.
compute PriorReportBeg = datesum(DtReportBeg,-6,'months').
execute.

if (dtpriordisch lt DtReportBeg AND dtpriordisch ge PriorReportBeg)
PDP6mo = 1.
execute.

if (PDP6mo eq 1 and duplicatechild eq 1 and missing(lag(DtDisch)))
flagPDerror eq 1.
execute.

compute dropforPDerror = 0.
if (flagPDerror eq 1 AND lag(entered) = 1) dropforPDerror = 1.
execute.

AGGREGATE outfile=* mode=ADDVARIABLES
/break childid
/dropchildPDE =max(dropforPDerror).

select if dropchildPDE ne 1.
execute.


* Select the records to keep.
select if Entered = 1.
execute.

*drop cases that discharge and reenter in same period.
if missing(flagfordeletion) flagfordeletion = 0.
execute.
select if flagfordeletion ne 1.
execute.

delete variables flagfordeletion.

************************************************
* If a child has more than one entry in the 12-month period, select only
the first episode.
```

```
************************************************

* We have already calculated time to reentry and put that in the child's
first episode, so dropping
* subsequent episodes at this point is okay.

* Flag the records to keep.
sort cases BY ChildID(A) DtLatRem(A).
match files
  /FILE=*
  /BY ChildID
  /FIRST=PrimaryFirst.
variable labels PrimaryFirst 'This episode reflects the childs first
entry into care during the specified FFY'.
value labels  PrimaryFirst 0 'Duplicate Case' 1 'Primary Case'.
variable level  PrimaryFirst (ORDINAL).
execute.

* Select the records to keep.
select if PrimaryFirst = 1.
execute.

delete variables PrimaryFirst.

************************************************
* REMOVE RECORDS WITH DQ PROBLEMS RELEVANT TO THIS MEASURE.
************************************************

* Do not delete dropped records if they occur in the last period needed
to calculate observed performance.
* This ensures we use data only from submissions required to observe the
cohort.
if DtReportEnd eq DtMeasureEnd DQ_Dropped = 0.
execute.

* Flag records with a problem (i.e., = 1) for any of the DQ checks used
for Perm in 12 (Entries) and Re-Entry,
* except DQ_IDNoMatchNext6Mo and DQ_totalrem1.
compute DQ_Indicator=0.
if any(1,DQ_DOBgtDtDisch, DQ_DOBgtDtLatRem, DQ_Dropped,
DQ_DtDischeqDtLatRem, DQ_DtDischltDtLatRem, DQ_gt21DOBtoDtDisch,
DQ_gt21DOBtoDtLatRem,  DQ_gt21DtDischtoDtLatRem, DQ_missDisreasn,
DQ_missDOB, DQ_missDtLatRem, DQ_dtpriordafterdlm)
DQ_Indicator=1.
execute.

* Delete records with a DQ problem(s).
select if DQ_Indicator=0.
execute.

* Delete the DQ variables as they are no longer needed.
delete variables DQ_Dropped to DQ_totalrem1 DQ_Indicator.
```

```
************************************************
CALCULATE LENGTH OF STAY
************************************************

* LOS (days) between date of latest removal from home and date of
discharge (or last date in the 3 yr period).
************************************************

compute LOSLatRemDays = $sysmis.
* For children who exited during the 3-yr period, LOS is time between
DtLatRem and DtDisch.
if not sysmis(DtDisch) LOSLatRemDays = datediff(DtDisch,DtLatRem,"days").
* For children who did not exit during the 3-yr period, LOS is time
between between DtLatRem and DtMeasureEnd.
if sysmis(DtDisch) LOSLatRemDays =
datediff(DtMeasureEnd,DtLatRem,"days").
* Trial Home Visit adjustment (adds 30 days to the LOS for children who
discharged to reunification whose most recent setting was THV).
if not(missing(DtCurSet)) and not(missing(DtLatRem)) and
(DtCurSet>DtLatRem) PreCurSetLOS=datediff(DtCurSet,DtLatRem, "days").
if (not(missing(DtDisch)) and (DtCurSet<DtDisch) and
not(missing(PreCurSetLOS)) and (disreasn=1) and curplset=8 and
datediff(DtDisch,DtCurSet,"days")>30) LOSLatRemDays=(PreCurSetLOS + 30).
execute.

* LOS (months) between date of latest removal from home and date of
discharge (or last date in the 3-yr period).
************************************************

compute LOSLatRemMo = $sysmis.
* For children who exited during the 3-yr period, LOS is time between
DtLatRem and DtDisch.
if not sysmis(DtDisch) LOSLatRemMo = datediff(DtDisch,DtLatRem,"months").
* For children who did not exit during the 3-yr period LOS is time
between DtLatRem and DtMeasureEnd.
if sysmis(DtDisch) LOSLatRemMo =
datediff(DtMeasureEnd,DtLatRem,"months").
* Trial Home Visit adjustment (adds "1" to month value (rather than "30"
to a day value).
if not(missing(DtCurSet)) and not(missing(DtLatRem)) and
(DtCurSet>DtLatRem) PreCurSetLOSmo=datediff(DtCurSet,DtLatRem, "months").
if (not(missing(DtDisch)) and (DtCurSet<DtDisch) and
not(missing(PreCurSetLOSmo)) and (disreasn=1) and curplset=8 and
datediff(DtDisch,DtCurSet,"days")>30) LOSLatRemMo=(PreCurSetLOSmo + 1).
execute.

* LOS (month categories) between date of latest removal from home and
date of discharge (or last date in the 3-yr period).
************************************************

recode LOSLatRemMo
(42 thru 251=8) (36 thru 42=7) (30 thru 36=6) (24 thru 30=5) (18 thru
24=4) (12 thru 18=3) (6 thru 12=2) (0 thru 6=1) into LOSLatRemMoCat.
```

```
* LOS is < 8 days.
if LOSLatRemDays < 8 LOSLatRemMoCat = 0.
execute.

* Formatting.
*************************************************

variable labels
LOSLatRemDays 'LOS - Days between DtLatRem and DtDisch (for discharged)
or last date in the 3-yr period (for still in care)'
LOSLatRemMo 'LOS - Months between DtLatRem and DtDisch (for discharged)
or last date in the 3-yr period (for still in care)'
LOSLatRemMoCat 'LOS - LOS in months, grouped into categories'.

value labels
LOSLatRemMoCat
0 '< 8 days'
1 '08 days – 5.99 mos'
2 '06 – 11.99 mos'
3 '12 – 17.99 mos'
4 '18 – 23.99 mos'
5 '24 – 29.99 mos'
6 '30 – 35.99 mos'
7 '36 – 41.99 mos'
8 '42 or more mos'.
formats LOSLatRemDays to LOSLatRemMoCat (F4.0).

delete variables PreCurSetLOS PreCurSetLOSmo.

*************************************************
CALCULATE IF CHILD EXITED TO PERM IN 12 MONTHS
*************************************************

* Flag episode if it should be included in the denominator and numerator
for Perm in 12 (Entries).
* Denominator: Child entered in the 12-month period (file already limited
to this group).
* Numerator (Num_Child = 1): Child in denominator exited to permanency in
12 months.

compute Den_Child=1.
compute Num_Child=$sysmis.
if DisReason2=1 and (LOSLatRemMoCat=1 or LOSLatRemMoCat=2) Num_Child = 1.
* If child exited to permanency but was 18 or older, this is considered a
failure so permanancy in 12 = 0.
if (Num_Child=1 and AgeXmosyrsCat=6) Num_Child = 0.
if (dtdisch gt Bday18) Num_Child = 0.
if missing(Num_Child) Num_Child = 0.
execute.

variable labels
Den_Child 'Perm in 12 (Entries) denominator - Child entered care in the
12-month period'
```

```
Num_Child 'Perm in 12 (Entries) numerator - Child exited to permanency
(all types) within 12 months of entering care'.
value labels Den_Child Num_Child
0 'No'
1 'Yes'.
execute.
formats Den_Child Num_Child (F1.0).

*************************************************
CALCULATE IF CHILD RE-ENTERED CARE IN 12 MONTHS
*************************************************

* Flag episode if it should be included in the denominator and numerator
for Re-Entry in 12.
* Denominator (DenRE_Child = 1): Child entered in the 12-month period
(file already limited to this group) AND
*      exited within 12 mos to reunification, LWR, or guardianship.
* Numerator (NumRE_Child = 1): Child in denominator re-entered in 12 mos
(i.e., TimeToNextReentry < 12).

if (DisReason1=1 or DisReason1=2 or DisReason1=5) and (LOSLatRemMoCat = 1
or LOSLatRemMoCat = 2) DenRE_Child=1.
if (DenRE_Child = 1 AND TimeToNextReentry < 12) NumRE_Child = 1.
if missing (DenRE_Child) DenRE_Child = 0.
if missing (NumRE_Child) NumRE_Child = 0.
execute.

variable labels
DenRE_Child 'ReEntry in 12 denominator - Child exited to reun, LWR, or
guardianship within 12 months of entering care'
NumRE_Child 'ReEntry in 12 numerator - Child exited to reun, LWR, or
guard within 12 months of entering care, AND re-entered care within 12
months'.
value labels DenRE_Child NumRE_Child
0 'No'
1 'Yes'.
execute.
formats DenRE_Child NumRE_Child (F1.0).

*************************************************
IDENTIFY AND REMOVE EPISODES THAT MEET EXCLUSION CRITERIA FOR PERM IN 12
*************************************************

* LOS < 8 days.
compute ExLOS8 = 0.
if LOSLatRemMoCat = 0 and not sysmis(DtDisch) ExLOS8 = 1.

* Age at entry is 18 or older.
compute ExAgeN18 = 0.
if AgeNmosyrsCat = 6 ExAgeN18 = 1.

variable labels
ExLOS8 'Episode has a LOS < 8 days'
ExAgeN18 'Child entered at age 18 or older'.
```

```
value labels ExLOS8
0 'No'
1 'Yes'
/ ExAgeN18
0 'No'
1 'Yes'.
execute.
formats ExLOS8 to ExAgeN18 (F1.0).

* Select the records to keep.
select if (ExLOS8 = 0 and ExAgeN18 = 0).
execute.

*************************************************
* Create variables holding state and national performance.
*************************************************

* State-level data.
sort cases by state (A) ChildID (A).
aggregate
  /OUTFILE=* MODE=ADDVARIABLES
  /PRESORTED
  /BREAK=state
  /Num_State=SUM(Num_Child)
  /Den_State=N.
compute Perf_State = Num_State / Den_State.
compute Perf_State_MP = (Num_State / Den_State) * 100.
execute.

* National-level data.
aggregate
  /OUTFILE=* MODE=ADDVARIABLES
  /BREAK=
  /Num_Nation=SUM(Num_Child)
  /Den_Nation=N.
compute Perf_Nation = Num_Nation / Den_Nation.
compute Perf_Nation_MP =  (Num_Nation / Den_Nation) * 100.
execute.
formats Perf_State Perf_Nation (F8.5).

variable labels
Den_State 'Perm in 12 (Entries) denominator - Number of children in state
who entered care in the 12-month period'
Num_State 'Perm in 12 (Entries) numerator - Among children in the
denominator, number of children in state who exited to permanency (all
types) within 12 months of entering care'
Perf_State 'Perm in 12 (Entries) - Percentage of children (entries) in
state who exited to permanency within 12 months of entering care'
Den_Nation 'Perm in 12 (Entries) denominator - Number of children in
nation who entered care in the 12-month period'
Num_Nation 'Perm in 12 (Entries) numerator - Among children in the
denominator, number of children in nation who exited to permanency (all
types) within 12 months of entering care'
```

```
Perf_Nation 'Perm in 12 (Entries) - Percentage of children (entries) in
nation who exited to permanency within 12 months of entering care'.

************************************************
CALCULATE STATE ENTRY RATES FOR THE 12-MONTH PERIOD
************************************************

* Entry rates are used as a risk adjustment variable for two indicators:
Permanency in 12 months for children
* entering and Re-entry to foster care in 12 months. This section of the
syntax uses the
* Child populations by state.sav file that was provided in the
CFSR3IndicatorSyntax.zip file, which can be updated
* with new annual estimates using the syntax, CFSR 3 - 0 Create census
child populations.sps.

* Each state's entry rate is calculated as the number of children in the
state entering foster care during the
* 12-month period divided by the number of children in the state’s child
population, multiplied by 1,000. The
* number of children entering foster care is as calculated in this
syntax, with all the data quality and other
* exclusions. The number of children in the state's child population is
obtained from the population division of the
* U.S. Census Bureau. This Census data reflect population estimates as of
July 1st of each year
* (POPULATION_YEAR), whereas the 12-month periods CB uses to define
children entering care are either
* October to September, or April to March. Therefore, we chose to use as
the denominator the Census year
* (July 1st, YYYY) closest to the 12-month period the child entered
foster care. For example, if the indicator
* follows children who entered care between April 1, 2011 and March 31,
2012 (an "11B/12A" file in AFCARS file
* conventions), we use child population estimates from the July 2011
Census year. If the 12-month period spanned
* October 1, 2012 through September 30, 2013, we would use population
estimates from the July 2013 Census
* year.

* Open the census data.
get file 'Fixed_FH\Child populations.sav'.
dataset name ChildPops window=front.

* Prevent accidentally overwriting child pop file.

* Merge in variables from Indicator file, then keep only DtPeriodBeg and
DtPeriodEnd.
match files
    /FILE=*
    /FILE='Indicator'
    /KEEP state to PopYear DtPeriodBeg DtPeriodEnd.
execute.
```

```
* Select the child populations from the year that falls between
DtPeriodBeg and DtPeriodEnd.
compute CensusYearApplies=0.
if (PopYear ge DtPeriodBeg) and (PopYear le DtPeriodEnd)
CensusYearApplies = 1.
EXECUTE.
select if censusyearapplies = 1.
execute.
sort cases by state.
execute.

* Merge in all variables from ChildPops files, then keep only
ChildPopulation.
dataset activate Indicator.
sort cases by state.
match files
    /FILE=*
    /TABLE='ChildPops'
    /BY state
    /KEEP state to Perf_Nation_MP ChildPopulation.
execute.
dataset close ChildPops.

* Calculate entry rate.
compute EntryRate = (Den_State/ChildPopulation) * 1000.
execute.

*************************************************
OUTPUT FILES
*************************************************

* Save.
save outfile='Observed_Child_FH\CFSR 3 - Observed perf for perm (entries)
' + PeriodType + YYstr + '.sav'
   /compressed.

* Save file for STATA (for multi-level modeling).
rename variables (AgeNmosyrs = ChildAge).
save translate OUTFILE='Observed_Child_FH\CFSR 3 - Observed perf for perm
(entries) ' + PeriodType + YYstr + '.dta'
  /TYPE=STATA
  /VERSION=8
  /EDITION=SE
  /MAP
  /REPLACE
  /KEEP=state stateabb statetxt TwelveMoCohort DtPeriodBeg DtPeriodEnd
ChildID Num_Child ChildAge Den_State
Num_State Perf_State Den_Nation Num_Nation Perf_Nation EntryRate.

* Change name back so we can use 'ChildAge" for the Re-Entry file.
rename variables (ChildAge = AgeNmosyrs).

* Create file with one record per state holding observed performance for
the 12-month cohort.
```

```
compute numstates = 1.
if (state eq lag(state))numstates = 0.
execute.

dataset copy OneRecordPerState.
dataset activate OneRecordPerState.
select if (numstates=1).
execute.
dataset activate OneRecordPerState.

string Indicator (A30).
execute.
compute Indicator = "Perm 12 (entries)".
execute.

save outfile='Observed_State_FH\CFSR 3 - Observed perf for perm (entries)
State file ' + PeriodType + YYstr + '.sav'
   /keep state stateabb statetxt Indicator TwelveMoCohort DtPeriodBeg
DtPeriodEnd Perf_State.

dataset activate Indicator.
dataset close OneRecordPerState.

*****************************************************
RE-ENTRY TO FOSTER CARE IN 12 MONTHS.
*****************************************************

dataset copy  ReEntry.
dataset activate ReEntry.
dataset close Indicator.

delete variables Num_State to Perf_Nation_MP.

*****************************************************
SELECT APPLICABLE RECORDS
*****************************************************

* Select only children in the denominator for re-entry.
*****************************************************

select if DenRE_Child=1.
execute.

*****************************************************
REMOVE EPISODES THAT MEET EXCLUSION CRITERIA FOR RE-ENTRY IN 12
*****************************************************

* Age at exit is 18 or older.
compute ExAgeX18 = 0.
if AgeXmosyrsCat = 6 ExAgeX18 = 1.
execute.

variable labels
ExAgeX18 'Child exited at age 18 or older'.
```

```
value labels ExAgeX18
0 'No'
1 'Yes'.
execute.
formats ExAgeX18 (F1.0).

* Select the records to keep.
select if (ExAgeX18 = 0).
execute.

* Create variables holding state and national performance (useful for
reporting).
**************************************************

* State-level data.
sort cases by state (A) ChildID (A).
aggregate
  /OUTFILE=* MODE=ADDVARIABLES
  /PRESORTED
  /BREAK=state
  /Num_State=SUM(NumRE_Child)
  /Den_State=N.
compute Perf_State = Num_State / Den_State.
compute Perf_State_MP = (Num_State / Den_State) * 100.
execute.

* National-level data.
aggregate
  /OUTFILE=* MODE=ADDVARIABLES
  /BREAK=
  /Num_Nation=SUM(NumRE_Child)
  /Den_Nation=N.
compute Perf_Nation = Num_Nation / Den_Nation.
compute Perf_Nation_MP =  (Num_Nation / Den_Nation) * 100.
execute.
formats Perf_State Perf_Nation (F8.5).

variable labels
Den_State 'ReEntry in 12 denominator - Number of children in state who
exited to reun, LWR, or guardianship within 12 months of entering care'
Num_State 'ReEntry in 12 numerator - Among children in the denominator,
number of children in state who re-entered care within 12 months'
Perf_State 'ReEntry in 12 - Percentage of children in state who re-
entered care within 12 months'
Den_Nation 'ReEntry in 12 denominator - Number of children in nation who
exited to reun, LWR, or guardianship within 12 months of entering care'
Num_Nation 'ReEntry in 12 numerator - Among children in the denominator,
number of children in nation who re-entered care within 12 months'
Perf_Nation 'ReEntry in 12 - Percentage of children in nation who re-
entered care within 12 months'.

**************************************************
OUTPUT FILES
**************************************************
```

```
* Save.
save outfile='Observed_Child_FH\CFSR 3 - Observed perf for reentry ' +
PeriodType + YYstr + '.sav'
   /compressed.

* Save file for STATA (for multi-level modeling).
rename variables (AgeXmosyrs = ChildAge).
save translate OUTFILE='Observed_Child_FH\CFSR 3 - Observed perf for
reentry ' + PeriodType + YYstr + '.dta'
  /TYPE=STATA
  /VERSION=13
  /EDITION=SE
  /MAP
  /REPLACE
  /KEEP=state stateabb statetxt TwelveMoCohort DtPeriodBeg DtPeriodEnd
ChildID /*recnumbr*/ NumRE_Child ChildAge Den_State
Num_State Perf_State Den_Nation Num_Nation Perf_Nation EntryRate.

rename variables (ChildAge = AgeXmosyrs).

* Create file with one record per state holding observed performance for
the 12-month cohort.
compute numstates = 1.
if (state eq lag(state))numstates = 0.
execute.

dataset copy OneRecordPerState.
dataset activate OneRecordPerState.
select if (numstates=1).
execute.
dataset activate OneRecordPerState.

string Indicator (A30).
execute.
compute Indicator = "Re-entry".
execute.

save outfile='Observed_State_FH\CFSR 3 - Observed perf for reentry State
file ' + PeriodType + YYstr + '.sav'
   /keep state stateabb statetxt Indicator TwelveMoCohort DtPeriodBeg
DtPeriodEnd Perf_State.

dataset activate ReEntry.
dataset close OneRecordPerState.

output save OUTFILE='SPSSLogs_FH\CFSR 3 - Observed perf for perm
(entries) & reentry ' + PeriodType + YYstr + '.spv'.

************************************************
ALL DONE.
************************************************
```

CFSR 3: OBSERVED PERFORMANCE - PERMANENCY IN 12 MONTHS FOR CHILDREN IN CARE ON FIRST DAY 12-23 MONTHS, and PERMANENCY IN 12 MONTHS FOR CHILDREN IN CARE ON FIRST DAY 24 MONTHS OR MORE

```
* Encoding: UTF-8.
set printback = on.

************************************************
CFSR 3 - CALCULATE OBSERVED PERFORMANCE
1. PERMANENCY IN 12 MONTHS FOR CHILDREN IN CARE ON FIRST DAY 12-23 MONTHS
2. PERMANENCY IN 12 MONTHS FOR CHILDREN IN CARE ON FIRST DAY 24 MONTHS OR
MORE.
************************************************

************************************************
GET SOURCE FILE
************************************************

*The file handle Root_FH represents the root folder for the CFSR
data.***** CHANGE THE ROOT FILE HANDLE TO MATCH THE WHERE YOU HAVE PLACED
THE CFSR 3 FOLDER ON YOUR COMPUTER***.
file handle Root_FH /name="C:\CFSR 3".

file handle AFCARS_Comb_FH  /name="Root_FH\0 - Source Data".

file handle AFCARS_DQ_FH /name= "Root_FH\2 - DQ AFCARS".

file handle SpssLogs_FH /name="Root_FH\5 - SPSS Logs".

file handle Observed_Child_FH /name="Root_FH\3 - Performance observed
child".

file handle Observed_State_FH /name="Root_FH\3 - Performance observed
state".

* Open source data.

GET file='AFCARS_Comb_FH\CFSR 3 AFCARS source data.sav'.
dataset name SourceData window=front.

* Prevent accidentally overwriting source data.
dataset copy  Indicator.
dataset activate Indicator.
dataset close SourceData.

* Delete unused variables.
delete variables amiakn asian blkafram hawaiipi white untodetm hisorgin
clindis mr vishear phydis dsmIII othermed everadpt ageadopt manrem
phyabuse sexabuse neglect aaparent
daparent aachild dachild childis chbehprb prtsdied prtsjail nocope
abandmnt relinqsh housing placeout casegoal ctkfamst ctk1yr ctk2yr
fosfamst fcctk1yr fcctk2yr rf1amakn
rf1asian rf1blkaa rf1nhopi rf1white rf1utod hofcctk1 rf2amakn rf2asian
rf2blkaa rf2nhopi rf2white rf2utod hofcctk2 ivefc iveaa ivaafdc ivdchsup
xixmedcd ssiother noa fcmntpay DtReview
DtLatRemTrans DtTPRMom DtTPRDad DtDischTrans.
```

```
************************************************
SPECIFY THE 12-MONTH COHORT WHOSE OUTCOME WILL BE ASSESSED
************************************************

* Identify the 12-month cohort of interest (e.g., children in care on the
first day of 13B14A)
* by entering "AB" or "BA," and the year the 12-month period ends.

* AB period (A file + B file) spans Oct 1 - Sept 30 of the following
year.
* BA period (B file + A file) spans Apr 1 - Mar 31 of the following year.

* Examples:
* ( * ) represents a FFY
* Children in care FD of    PeriodType  YYYY   YYstr
* 09B10A                        BA          2010     10
* 10A10B *                      AB          2010     10
* 10B11A                        BA          2011     11
* 11A11B *                      AB          2011     11
* 11B12A                        BA          2012     12
* 12A12B *                      AB          2012     12
* 12B13A                        BA          2013     13
* 13A13B *                      AB          2013     13
* 13B14A                        BA          2014     14
* etc.

********  START USER INPUT ********

define PeriodType () "XX"
!enddefine.
define YYYY () XXXX.
!enddefine.
define YYstr () "XX"
!enddefine.

********  END USER INPUT ********

* Create variable that indicates the user-specified 12-month cohort
(e.g., "BA14" represents "13B14A").
string TwelveMoCohort (A4).
compute TwelveMoCohort = concat(PeriodType,YYstr).
execute.

************************************************
CREATE DATE PARAMETERS
************************************************

* Start date of a 12-month period (10/1/YYYY or 4/1/YYYY).
* End date of a 12-month period (3/31/YYYY or 9/30/YYYY).

* Perm by 12 (FD) involves following children who were in care on the
first day of the 12-month period specified
* earlier. Identify the start date (DtPeriodBeg) and end date
(DtPeriodEnd) that defines this 12-month period.
```

```
do if PeriodType = "AB".
compute DtPeriodBeg=date.mdy(10,01,YYYY-1).
compute DtPeriodEnd=date.mdy(09,30,YYYY).
compute DtEndFirst6mo=date.mdy(03,31,YYYY).
end if.
do if PeriodType = "BA".
compute DtPeriodBeg=date.mdy(04,01,YYYY-1).
compute DtPeriodEnd=date.mdy(03,31,YYYY).
compute DtEndFirst6mo=date.mdy(9,30,YYYY-1).
end if.
execute.
formats DtPeriodBeg DtPeriodEnd DtEndFirst6mo (adate10).

variable labels
TwelveMoCohort '12-month period of which children were in care on the 1st
day for whom performance is being assessed'
DtPeriodBeg 'Start date of the first 12-month period specified'
DtPeriodEnd 'End date of the first 12-month period specified'.

************************************************
* Select only 6-month records that were reported during the 12-month
period
* (between DtPeriodBeg and DtPeriodEnd).
************************************************

* Flag the records to keep.
if (DtReportBeg ge DtPeriodBeg) and (DtReportEnd le DtPeriodEnd)
ReportedDuringPeriod = 1.
execute.

* Select the records to keep.
select if ReportedDuringPeriod=1.
execute.

delete variables ReportedDuringPeriod.

************************************************
REMOVE STATES THAT EXCEED DQ LIMITS
************************************************

* The AFCARS cross-file checks (Dropped records and AFCARS IDs don't
match from one period to the next)
* apply only to the first 6-month period (DtReportBeg = DtPeriodBeg). The
AFCARS within-file checks apply to
* both periods (DtReportBeg ge DtPeriodBeg AND DtReportBeg le
DtPeriodEnd).

* Open the data quality results done previously.
* Open the data quality results done previously.
get file= 'AFCARS_DQ_FH\Merged AFCARS DQ files.sav'.
dataset name DQResults window=front.

* Prevent accidentally overwriting CFSR 3 data quality results file.
```

```
dataset copy  DQIndicator.
dataset activate DQIndicator.
dataset close DQResults.

* Merge in variables from Indicator file, then keep only DtPeriodBeg
DtPeriodEnd.

do if PeriodType = "AB".
compute DtPeriodBeg=date.mdy(10,01,YYYY-1).
compute DtPeriodEnd=date.mdy(09,30,YYYY).
compute DtEndFirst6mo=date.mdy(03,31,YYYY).
end if.
do if PeriodType = "BA".
compute DtPeriodBeg=date.mdy(04,01,YYYY-1).
compute DtPeriodEnd=date.mdy(03,31,YYYY).
compute DtEndFirst6mo=date.mdy(09,30,YYYY-1).
end if.
execute.
formats DtPeriodBeg DtPeriodEnd (adate10).

variable labels
DtPeriodBeg 'Start date of the first 12-month period specified'
DtPeriodEnd 'End date of the first 12-month period specified'.

* Select the AFCARS checks for this indicator and DQ results for the 6-
month period(s).
* ... Cross file checks (apply only to the first 6-month period).
if PermReEntry = 1 and DQFileXW = "Cross file" AND (DtReportBeg eq
DtPeriodBeg) DQChecksApply = 1.
* ... Within file checks (apply to both 6-month periods).
if PermReEntry and DQFileXW = "Within file" AND (DtReportBeg ge
DtPeriodBeg) and (DtReportEnd le DtPeriodEnd) DQChecksApply = 1.
execute.

* For each state, for the checks that apply, sum the number of checks it
failed.
sort cases by state.
select if DQChecksApply = 1.
execute.
dataset declare DQstate.
aggregate
  /OUTFILE=DQState
  /BREAK=state
  /DQFailedCheck_sum=SUM(DQFailedCheck).

* Merge states' DQFailedCheck_sum value into Indicator file. Any state
where DQFailedCheck_sum > 0 will be
* dropped from the dataset and excluded from analyses.

* ... Merge in all variables from DQIndicator file, then keep only
DQFailedCheck_sum.
dataset activate Indicator.
sort cases by state.
match files
```

```
    /FILE=*
    /TABLE='DQState'
    /BY state
    /KEEP state to DtEndFirst6mo DQFailedCheck_sum.
execute.
dataset close DQIndicator.
dataset close DQstate.

* Select only states that met the DQ checks.
select if DQFailedCheck_sum = 0.
execute.
delete variables DQFailedCheck_sum.

* Count the number of states remaining.
compute numstates = 1.
if (state eq lag(state))numstates = 0.
EXECUTE.

*************************************************
SELECT APPLICABLE RECORDS
*************************************************

* Select only the most recent 6-month record reported for each child, for
each episode.
*************************************************

* This will create an episode-level file. Episodes are distinguished by
the date of latest removal from home.
* A child has a record in each 6-month submission until he discharged,
dropped, or was still in care as of
* the last reporting period we have in the file. We only want one record
per child, per episode, and the record we pick
* is the last one reported for that episode. This record will contain the
most recent data available that describes
* the child's episode (e.g., LOS, age on first day, etc.).

* Flag the records to keep.
sort cases BY ChildID (A) DtLatRem (A).
match files
  /FILE=*
  /BY ChildID DtLatRem
  /LAST=Flag_LastRpt4Ep.
variable labels Flag_LastRpt4Ep 'last 6-month report we received for this
episode (based on DtLatRem)'.
value labels Flag_LastRpt4Ep 0 'Duplicate Case' 1 'Primary Case'.
variable level Flag_LastRpt4Ep (ORDINAL).
execute.

* Select the last 6-month record for each child, for each episode.
select if Flag_LastRpt4Ep=1.
execute.

delete variables Flag_LastRpt4Ep.
```

```
* Select only children in care on the first day of the 12-month period
specified earlier.
************************************************
* This is the cohort of children for whom performance will be examined.

* Flag the records to keep.

*flag if record is for a child's second episode in the analysis period.
sort cases by  childid dtreportbeg.
EXECUTE.
if (childid eq lag(childid)) duplicate_child=1.
execute.

*label each unique episode for a child.
compute position=1.
if (duplicate_child=1) position = lag(position)+1.
execute.

*create child level flag that indicates if child has more than one
episode.
aggregate
outfile=* mode=ADDVARIABLES
/break state childid
/dupchild = max(duplicate_child).

*create holding variable for dtpriordisch from subsequent record.

PRESERVE.
set workspace 200000.
create TempDtPriorDisch=Lead(DtPriorDisch,1).
execute.
RESTORE.

*Create varaible for usable dtpriordisch when child has more than one
episode and the record is for the child's first episode in the analysis
period. Create flag for when the date reported in dtpriordisch
*falls in the first 6-months of the analysis period.
compute ExMissingDOD=0.
EXECUTE.

if dupchild=1 and position=1 and missing(DtDisch)
DtDischNextRecord=TempDtPriorDisch.
if DtDischNextRecord ge DtPeriodBeg and DtDischNextRecord le
DtEndFirst6Mo ExMissingDOD=1.
execute.

if (DtDischNextRecord ge DtEndFirst6Mo) FlagforRemove =1.
execute.

if missing(FlagforRemove) FlagforRemove=0.
execute.
select if FlagforRemove =0 AND ExMissingDOD eq 0.
execute.
```

```
DELETE VARIABLES TempDtPriorDisch DtDischNextRecord.

*flag (InAtStart) and keep only records where the child was in foster
care at the start of the analysis period.
if ((DtLatRem lt DtPeriodBeg) and (DtDisch ge (DtPeriodBeg) |
missing(DtDisch))) InAtStart = 1.
recode InAtStart (sysmis = 0).
execute.

variable labels InAtStart 'Child was in care on the first day of the
specified 12-month period'.
value labels InAtStart
0 'No'
1 'Yes'.
EXECUTE.
* Select the records to keep.
select if InAtStart = 1.
execute.

*************************************************
* REMOVE RECORDS WITH DQ PROBLEMS RELEVANT TO THIS MEASURE.
*************************************************

* Do not delete dropped records if they occur in the last period needed
to calculate observed performance.
* This ensures we use only data from submissions required to observe the
cohort.
if DtReportEnd eq DtPeriodEnd DQ_Dropped = 0.
execute.

* Flag records with a problem (i.e., = 1) for any of the DQ checks used
for Perm by 12 (FD),
* except DQ_IDNoMatchNext6Mo and DQ_totalrem1.
compute DQ_Indicator=0.
if any(1,DQ_Dropped, DQ_missDOB, DQ_missDtLatRem, DQ_DOBgtDtLatRem,
DQ_DOBgtDtDisch, DQ_gt21DOBtoDtLatRem, DQ_gt21DOBtoDtDisch,
DQ_gt21DtDischtoDtLatRem, DQ_DtDischeqDtLatRem, DQ_DtDischltDtLatRem,
DQ_missDisreasn, DQ_dtpriordafterdlm)
DQ_Indicator=1.
execute.

* Delete records with a DQ problem(s).
select if DQ_Indicator=0.
execute.

* Delete the DQ variables as they are no longer needed.
delete variables DQ_Dropped to DQ_totalrem1 DQ_Indicator.

*************************************************
CALCULATE AGE ON FIRST DAY
*************************************************

* Age on first day in years.
compute AgeFDyrs = datediff(DtPeriodBeg,DtBirth,"years").
```

```
* Recode age in years to desired categories.
recode AgeFDyrs (0=0) (1 thru 5=1) (6 thru 10=2) (11 thru 16=3) (17=4)
(18 thru 21=5) (else=sysmis) into AgeFDyrsCat.
execute.

* Handle problem values.
*******************************
If age at exit can't be calculated, recode to 9999 (missing DOB).

do if DtBirth > DtPeriodBeg.
recode AgeFDyrs AgeFDyrsCat (else = 8888).
end if.
* If time between date of birth and date of first day is > 21 yrs (i.e.,
age on FD is > 21 yrs), recode to 8888.
compute #AgeFD = datediff(DtPeriodBeg,DtBirth,"years").
do if #AgeFD > 21.
recode AgeFDyrs AgeFDyrsCat (else = 8888).
end if.

missing values AgeFDyrs AgeFDyrsCat (8888, 9999).

* Formatting.
************************************************

variable labels
AgeFDyrs 'Age on first day in years (1, 2, 3 ... 21)'
AgeFDyrsCat 'Age on first day in years in categores (1-5, 6-10, etc...)'.

* Create macro that holds age values.
DEFINE !AgeYrs ( )
0 '< 1 yr'
1 '1 yr'
2 '2 yrs'
3 '3 yrs'
4 '4 yrs'
5 '5 yrs'
6 '6 yrs'
7 '7 yrs'
8 '8 yrs'
9 '9 yrs'
10 '10 yrs'
11 '11 yrs'
12 '12 yrs'
13 '13 yrs'
14 '14 yrs'
15 '15 yrs'
16 '16 yrs'
17 '17 yrs'
18 '18 yrs'
19 '19 yrs'
20 '20 yrs'
21 '21 yrs'
!ENDDEFINE.
```

```
DEFINE !AgeYrsCat ( )
0 '< 1 yr'
1 '01 - 05'
2 '06 - 10'
3 '11 - 16'
4 '17'
5 '18 - 21'
!ENDDEFINE.

value labels
AgeFDyrs
!AgeYrs
8888 'Invalid (age is negative or > 21)'
9999 'Cannot calculate (DOB missing)'
/ AgeFDyrsCat
!AgeYrsCat
8888 'Invalid (age is negative or > 21)'
9999 'Cannot calculate (DOB missing)'.

formats AgeFDyrs to AgeFDyrsCat (F2.0).

************************************************
CALCULATE LENGTH OF STAY FROM FIRST DAY FORWARD
************************************************

* Note - Some children may have dates of discharge that occur after the
last reporting period in the file,
* and therefore LOS > 12 months. These are most likely due to states
being able to submit data up to
* 45 days after the end of the reporting period, in which they may
include discharges that occur within
* that 45 day timeframe.

* LOS (days) between first day of the year and date of discharge (or last
date in the 12-month period).
*********************************************************************
*****************************************************************
compute LOSFDDays = $sysmis.
 * For children who exited after the FD, LOS is time between DtPeriodBeg
and DtDisch.
if not sysmis(DtDisch) LOSFDDays = datediff(DtDisch,DtPeriodBeg,"days").
 * For children who did not exit during the 12-month period, LOS is time
between between DtPeriodBeg and DtPeriodEnd.
do if sysmis(DtDisch).
compute LOSFDDays = datediff(DtPeriodEnd,DtPeriodBeg,"days") + 1. /*
Since datediff is not inclusive of the last date, we have to add in one
day to the length of stay to get 365 (or 366 for children in care the
entire period) .
end if.
execute.

if (dtdisch gt dtdischadjusted) dischafter18 eq 1.
execute.
```

```
 * LOS (months) between first day of the year and date of discharge (or
last date in the 12-month period).
************************************************

compute LOSFDMo = $sysmis.
 * For children who exited during the 12-month period, LOS is time
between DtPeriodBeg and DtDisch.
if not sysmis(DtDisch) LOSFDMo = datediff(DtDisch,DtPeriodBeg,"months").
 * For children who did not exit during the 12-month period, LOS is time
between between DtPeriodBeg and DtPeriodEnd.
if sysmis(DtDisch) LOSFDMo = datediff(DtPeriodEnd,DtPeriodBeg,"months").

execute.


 * LOS (month categories) between first day of the year and date of
discharge (or last date in the 12-month period).
************************************************

recode LOSFDMo
 (12 thru 251=3) (6 thru 12=2) (0 thru 6=1) into LOSFDMoCat.


 * Formatting.
************************************************

variable labels
LOSFDdays 'LOS - Days between DtPeriodBeg (i.e., FD) and DtDisch (for
discharged) or last date in the 12-month period (for still in care)'
LOSFDMo 'LOS - Months between DtPeriodBeg (i.e., FD) and DtDisch (for
discharged) or last date in the 12-month period  (for still in care)'
LOSFDMoCat 'LOS in months, grouped into categories'.

value labels
LOSFDMoCat
1 'Less than 6 mos'
2 '06 - 11.99 mos'
3 '12 or more mos'.

 * formats LOSFDDays to LOSFDMoCat (F4.0).

************************************************
CALCULATE TIME IN CARE PRIOR and UP TO FIRST DAY
************************************************

* This is used to identify children who, as of the first date, have been
in care 12-23 months from children in care
* 24 months or more. LOSPriorFDMo = Prior time in care.

if not(missing(DtLatRem))
LOSPriorFDMo=datediff(DtPeriodBeg,DtLatRem,"months").
execute.
recode LOSPriorFDMo
```

```
(Lowest thru 5.99=1) (6 thru 11.99=2) (12 thru 23.99=3) (24 thru
251.99=4) INTO LOSPriorFDMoCat.
execute.

value labels LOSPriorFDMoCat
1 'Under 6 months'
2 '6 to 11 mos'
3 '12 to 23 mos'
4 '2 years or longer' .

formats LOSFDMo LOSPriorFDMoCat (F4.0).

*************************************************
CALCULATE TIME IN CARE FROM LATEST REMOVAL TO DISCHARGE
*************************************************

* Later, we want to identify and exclude children who entered prior to
the first day and exited
* within 8 days (i.e., LOS < 8 days).
if not(sysmis(DtDisch)) AND Dtdisch ge dtperiodbeg AND Dtdisch le
dtperiodend LOSLatRemDays = datediff(DtDisch,DtLatRem,"days").
execute.

*************************************************
CALCULATE IF CHILD EXITED TO PERM BY 12 MONTHS
*************************************************

* Flag episode if it should be included in the numerator.
* Denominator: Child was in care on the first day of the 12-month period
(file already limited to this group).
* Numerator (Num_Child = 1): Child in denominator exited to permanency by
12 months.

compute Num_Child=$sysmis.
if DisReason2=1 and (LOSFDMoCat=1 or LOSFDMoCat=2) AND (dtdisch ge
dtperiodbeg AND dtdisch le dtperiodend)  Num_Child = 1.
* If child exited to permanency but was 18 or older, this is considered a
failure so permanancy by 12 = 0.
if (dtdisch gt Bday18) Num_Child = 0.
if missing(Num_Child) Num_Child = 0.
execute.

variable labels Num_Child 'Child exited to permanency (all types) within
12 months of the first day'.
value labels Num_Child
0 'No'
1 'Yes'.
execute.
formats Num_Child (F1.0).

*************************************************
REMOVE EPISODES THAT MEET EXCLUSION CRITERIA
*************************************************
```

```
* LOS < 8 days (LOS based on latest removal - NOT LOS from first day).
compute ExLOS8 = 0.
if (LOSLatRemDays lt 8) and not sysmis(DtDisch) ExLOS8 = 1.

* Age on first day 18 or older.
compute ExAgeFD18 = 0.
if AgeFDyrsCat = 5 ExAgeFD18=1.
* Age on first day is invalid or can't be calculated.
if missing(AgeFDyrsCat) ExAgeFD18 = 9999.
execute.

variable labels
ExLOS8 'Episode has a LOS < 8 days'
ExAgeFD18 'Child was age 18 or older on first day'.

value labels ExLOS8
0 'No'
1 'Yes'
9999 'Cannot be determined because LOS is negative, > 21, or could not be
calculated due to missing DtLatRem'
/ ExAgeFD18
0 'No'
1 'Yes'
9999 'Cannot be determined because age is negative, > 21, or could not be
calculated due to missing DOB or DtLatRem)'.
execute.

formats ExLOS8 to ExAgeFD18 (F1.0).
MISSING VALUES ExLOS8 to ExAgeFD18 (8888 to 9999).
* Report number and % of episodes that will be excluded, by state.
EXECUTE.

* Select the records to keep (i.e., 0 for both vars.
select if (ExLOS8 = 0 and ExAgeFD18 = 0).
execute.

************************************************
* SUMMARY STATISTICS
* PERMANENCY IN 12 MONTHS FOR CHILDREN IN CARE ON FIRST DAY 12-23 MONTHS
************************************************

* Make a copy of dataset which right now includes both 12-23 and 2+years
cohorts.
dataset copy FD_1223.
dataset activate FD_1223.

* Select the 12-24 cohort.
select if LOSPriorFDMoCat = 3.
execute.

* Create variables holding state and national performance.
************************************************

* State-level data.
```

```
sort cases by state (A) ChildID (A).
aggregate
  /OUTFILE=* MODE=ADDVARIABLES
  /PRESORTED
  /BREAK=state
  /Num_State=SUM(Num_Child)
  /Den_State=N.
compute Perf_State = Num_State / Den_State.
compute Perf_State_MP = (Num_State / Den_State) * 100.
execute.

* National-level data.
aggregate
  /OUTFILE=* MODE=ADDVARIABLES
  /BREAK=
  /Num_Nation=SUM(Num_Child)
  /Den_Nation=N.
compute Perf_Nation = Num_Nation / Den_Nation.
compute Perf_Nation_MP = (Num_Nation / Den_Nation) * 100.
execute.
formats Num_State Num_Nation (F8.0) Perf_State Perf_Nation (F8.4).

variable labels
Den_State 'Perm in 12 (12-23) denominator - Number of children in state
in care on the first day of the 12-month period, who as of the first day
had been in care 12-23 months'
Num_State 'Perm in 12 (12-23) numerator - Among children in the
denominator, number of children who exited to permanency (all types)
within 12 months of the first day'
Perf_State 'Perm in 12 (12-23) - Percentage of children in care on the
first day (12-23 months) who exited to permanency within 12 months of the
first day'
Den_Nation 'Perm in 12 (12-23) denominator - Number of children in nation
in care on the first day of the 12-month period, who as of the first day
had been in care 12-23 months'
Num_Nation 'Perm in 12 (12-23) numerator - Among children in the
denominator, number of children in nation who exited to permanency (all
types) within 12 months of the first day'
Perf_Nation 'Perm in 12 (12-23) - Percentage of children in care on the
first day (12-23 months) who exited to permanency within 12 months of the
first day'.

*************************************************
* OUTPUT FILES
*************************************************

* Save.
save outfile='Observed_Child_FH\CFSR 3 - Observed perf for perm (FD 12-
23) ' + PeriodType + YYstr + '.sav'
   /compressed.

* Save file for STATA (for multi-level modeling).
rename variables (AgeFDyrs = ChildAge).
```

```
save translate OUTFILE='Observed_Child_FH\CFSR 3 - Observed perf for perm
(FD 12-23) ' + PeriodType + YYstr + '.dta'
  /TYPE=STATA
  /VERSION=8
  /EDITION=SE
  /MAP
  /REPLACE
  /KEEP=state stateabb statetxt TwelveMoCohort DtPeriodBeg DtPeriodEnd
ChildID Num_Child ChildAge Den_State
Num_State Perf_State Den_Nation Num_Nation Perf_Nation.
rename variables (ChildAge = AgeFDyrs).

* Save file with one record per state holding observed performance for
the 12-month cohort.
compute numstates = 1.
if (state eq lag(state))numstates = 0.
execute.

dataset copy OneRecordPerState.
dataset activate OneRecordPerState.
select if (numstates=1).
execute.
dataset activate OneRecordPerState.

string Indicator (A30).
execute.
compute Indicator = "Perm 12 (FD 12-23)".
execute.

save outfile= 'Observed_State_FH\CFSR 3 - Observed perf for perm (FD 12-
23) State file ' + PeriodType + YYstr + '.sav'
   /keep state stateabb statetxt Indicator TwelveMoCohort DtPeriodBeg
DtPeriodEnd Perf_State.

dataset activate FD_1223.
dataset close OneRecordPerState.

************************************************
* SUMMARY STATISTICS
* PERMANENCY IN 12 MONTHS FOR CHILDREN IN CARE ON FIRST DAY 24 MONTHS OR
MORE
************************************************

dataset activate Indicator.
dataset close FD_1223.

* Make a copy of dataset which right now includes both 12-23 and 2+years
cohorts.
dataset copy FD_24Plus.
dataset activate FD_24Plus.

* Select the 24 months and more cohort.
select if LOSPriorFDMoCat = 4.
execute.
```

```
* Create variables holding state and national performance (useful for
reporting).
************************************************

* State-level data.
sort cases by state (A) ChildID (A).
aggregate
  /OUTFILE=* MODE=ADDVARIABLES
  /PRESORTED
  /BREAK=state
  /Num_State=SUM(Num_Child)
  /Den_State=N.
compute Perf_State = Num_State / Den_State.
compute Perf_State_MP = (Num_State / Den_State) * 100.
execute.

* National-level data.
aggregate
  /OUTFILE=* MODE=ADDVARIABLES
  /BREAK=
  /Num_Nation=SUM(Num_Child)
  /Den_Nation=N.
compute Perf_Nation = Num_Nation / Den_Nation.
compute Perf_Nation_MP = (Num_Nation / Den_Nation) * 100.
execute.
formats Num_State Num_Nation (F8.0) Perf_State Perf_Nation (F8.4).

variable labels
Den_State 'Perm in 12 (24 or more) denominator - Number of children in
state in care on the first day of the 12-month period, who as of the
first day had been in care 24 months or more'
Num_State 'Perm in 12 (24 or more) numerator - Among children in the
denominator, number of children who exited to permanency (all types)
within 12 months of the first day'
Perf_State 'Perm in 12 (24 or more) - Percentage of children in care on
the first day (24 months or more) who exited to permanency within 12
months of the first day'
Den_Nation 'Perm in 12 (24 or more) denominator - Number of children in
nation in care on the first day of the 12-month period, who as of the
first day had been in care 24 months or more'
Num_Nation 'Perm in 12 (24 or more) numerator - Among children in the
denominator, number of children in nation who exited to permanency (all
types) within 12 months of the first day'
Perf_Nation 'Perm in 12 (24 or more) - Percentage of children in care on
the first day (24 months or more) who exited to permanency within 12
months of the first day'.
```

```
*************************************************
OUTPUT FILES
*************************************************

* Save.
save outfile='Observed_Child_FH\CFSR 3 - Observed perf for perm (FD 24 or
more) ' + PeriodType + YYstr + '.sav'
   /compressed.

* Save file for STATA (for multi-level modeling).
rename variables (AgeFDyrs = ChildAge).
save translate OUTFILE='Observed_Child_FH\CFSR 3 - Observed perf for perm
(FD 24 or more) ' + PeriodType + YYstr + '.dta'
  /TYPE=STATA
  /VERSION=8
  /EDITION=SE
  /MAP
  /REPLACE
  /KEEP=state stateabb statetxt TwelveMoCohort DtPeriodBeg DtPeriodEnd
ChildID Num_Child ChildAge Den_State
Num_State Perf_State Den_Nation Num_Nation Perf_Nation.
rename variables (ChildAge = AgeFDyrs).

* Save file with one record per state holding observed performance for
the 12-month cohort.
compute numstates = 1.
if (state eq lag(state))numstates = 0.
execute.

dataset copy OneRecordPerState.
dataset activate OneRecordPerState.
select if (numstates=1).
execute.
dataset activate OneRecordPerState.

string Indicator (A30).
execute.
compute Indicator = "Perm 12 (FD 24 or more)".
execute.

save outfile='Observed_State_FH\CFSR 3 - Observed perf for perm (FD 24 or
more) State file ' + PeriodType + YYstr + '.sav'
   /keep state stateabb statetxt Indicator TwelveMoCohort DtPeriodBeg
DtPeriodEnd Perf_State.

dataset activate FD_24Plus.
dataset close OneRecordPerState.

output save OUTFILE='SpssLogs_FH\CFSR 3 - Observed perf for perm (FD) ' +
PeriodType + YYstr + '.spv'.

*************************************************************
ALL DONE.
*************************************************************
```

CFSR 3: OBSERVED PERFORMANCE - PLACEMENT STABILITY

```
* Encoding: UTF-8.

************************************************
CFSR 3 - CALCULATE OBSERVED PERFORMANCE
- PLACEMENT STABILITY
************************************************

*The file handle Root_FH represents the root folder for the CFSR data.
***** CHANGE THE ROOT FILE HANDLE TO MATCH THE WHERE YOU HAVE PLACED THE
CFSR 3 FOLDER ON YOUR COMPUTER***.
file handle Root_FH /name="C:\CFSR 3".

file handle AFCARS_Comb_FH  /name="Root_FH\0 - Source Data".

file handle AFCARS_DQ_FH /name= "Root_FH\2 - DQ AFCARS".

file handle SpssLogs_FH /name="Root_FH\5 - SPSS Logs".

file handle Observed_Child_FH /name="Root_FH\3 - Performance observed
child".

file handle Observed_State_FH /name="Root_FH\3 - Performance observed
state".

* Open source data.

GET file='AFCARS_Comb_FH\CFSR 3 AFCARS source data.sav'.
dataset name SourceData window=front.
cache.
* Prevent accidentally overwriting source data.
dataset copy  Indicator.
dataset activate Indicator.
dataset close SourceData.

* Delete unused variables.

delete variables amiakn asian blkafram hawaiipi white untodetm hisorgin
clindis mr vishear phydis dsmIII othermed everadpt ageadopt manrem
phyabuse sexabuse neglect aaparent
daparent aachild dachild childis chbehprb prtsdied prtsjail nocope
abandmnt relinqsh housing placeout casegoal ctkfamst ctk1yr ctk2yr
fosfamst fcctk1yr fcctk2yr rf1amakn
rf1asian rf1blkaa rf1nhopi rf1white rf1utod hofcctk1 rf2amakn rf2asian
rf2blkaa rf2nhopi rf2white rf2utod hofcctk2 ivefc iveaa ivaafdc ivdchsup
xixmedcd ssiother noa fcmntpay DtReview
DtLatRemTrans DtTPRMom DtTPRDad DtDischTrans.

************************************************
SPECIFY THE 12-MONTH COHORT WHOSE OUTCOME WILL BE ASSESSED
************************************************

* Identify the 12-month cohort of interest (e.g., children entering in
13B14A)
```

```
* by entering "AB" or "BA" for PeriodType and the year the 12-month
period ends (for YYYY and YYstr).

* AB period (A file + B file) spans Oct 1 - Sept 30 of the following
year.
* BA period (B file + A file) spans Apr 1 - Mar 31 of the following year.

* Examples:
* ( * ) represents a FFY
* Children who entered     PeriodType  YYYY   YYstr
* 09B10A                        BA              2010     10
* 10A10B *                      AB              2010     10
* 10B11A                        BA              2011     11
* 11A11B *                      AB              2011     11
* 11B12A                        BA              2012     12
* 12A12B *                      AB              2012     12
* 12B13A                        BA              2013     13
* 13A13B *                      AB              2013     13
* 13B14A                        BA              2014     14
* etc.

********  START USER INPUT ********

define PeriodType () "XX"
!enddefine.
define YYYY () XXXX.
!enddefine.
define YYstr () "XX"
!enddefine.

********  END USER INPUT ********

*************************************************
CREATE DATE PARAMETERS
*************************************************

* Start date of a 12-month period (10/1/YYYY or 4/1/YYYY).
* End date of a 12-month period (3/31/YYYY or 9/30/YYYY).

* Placement stability involves following children who entered during the
12-month period specified earlier.
* Identify the start date (DtPeriodBeg) and end date (DtPeriodEnd) that
defines this 12-month period.

do if PeriodType = "AB".
compute DtPeriodBeg=date.mdy(10,01,YYYY-1).
compute DtPeriodEnd=date.mdy(09,30,YYYY).
compute DtEndFirst6mo=date.mdy(03,31,YYYY).
end if.
do if PeriodType = "BA".
compute DtPeriodBeg=date.mdy(04,01,YYYY-1).
compute DtPeriodEnd=date.mdy(03,31,YYYY).
compute DtEndFirst6mo=date.mdy(09,30,YYYY-1).
end if.
```

```
execute.
formats DtPeriodBeg DtPeriodEnd (adate10).

variable labels
DtPeriodBeg 'Start date of the first 12-month period specified'
DtPeriodEnd 'End date of the first 12-month period specified'.

if (DtReportBeg ge DtPeriodBeg) and (DtReportEnd le DtPeriodEnd)
ReportedDuringPeriod = 1.
execute.

* Select the records to keep.
select if ReportedDuringPeriod=1.
execute.

delete variables ReportedDuringPeriod.

* Create variable that indicates the user-specified 12-month cohort
(e.g., "BA14" represents "13B14A").
string TwelveMoCohort (A4).
compute TwelveMoCohort = concat(PeriodType,YYstr).
execute.

*************************************************
REMOVE STATES THAT EXCEED DQ LIMITS
*************************************************

* The AFCARS cross-file checks (Dropped records and AFCARS IDs don't
match from one period to the next)
* apply only to the first 6-month period (DtReportBeg = DtPeriodBeg). The
AFCARS within-file checks apply to
* both periods (DtReportBeg ge DtPeriodBeg AND DtReportBeg le
DtPeriodEnd).

* Open the data quality results done previously.
get file= 'AFCARS_DQ_FH\Merged AFCARS DQ files.sav'.
dataset name DQResults window=front.

* Prevent accidentally overwriting CFSR 3 data quality results file.
dataset copy  DQIndicator.
dataset activate DQIndicator.
dataset close DQResults.

*************************************************
CREATE DATE PARAMETERS (In The DQ File)
*************************************************

do if PeriodType = "AB".
compute DtPeriodBeg=date.mdy(10,01,YYYY-1).
compute DtPeriodEnd=date.mdy(09,30,YYYY).
compute DtEndFirst6mo=date.mdy(03,31,YYYY).
end if.
do if PeriodType = "BA".
compute DtPeriodBeg=date.mdy(04,01,YYYY-1).
```

```
compute DtPeriodEnd=date.mdy(03,31,YYYY).
compute DtEndFirst6mo=date.mdy(09,30,YYYY-1).
end if.
execute.
formats DtPeriodBeg DtPeriodEnd (adate10).

variable labels
DtPeriodBeg 'Start date of the first 12-month period specified'
DtPeriodEnd 'End date of the first 12-month period specified'.

* Select the AFCARS checks for this indicator and DQ results for the 6-
month period(s).
* ... Cross file checks (apply only to the first 6-month period).
if PS = 1 and DQFileXW = "Cross file" AND (DtReportBeg eq DtPeriodBeg)
DQChecksApply = 1.
* ... Within file checks (apply to both 6-month periods).
if PS = 1 and DQFileXW = "Within file" AND (DtReportBeg ge DtPeriodBeg)
and (DtReportEnd le DtPeriodEnd) DQChecksApply = 1.
execute.

* For each state, for the checks that apply, sum the number of checks it
failed.
sort cases by state.
select if DQChecksApply = 1.
execute.
dataset declare DQstate.
aggregate
  /OUTFILE=DQState
  /BREAK=state
  /DQFailedCheck_sum=SUM(DQFailedCheck).

execute.

* ... Merge in all variables from DQIndicator, then keep only
DQFailedCheck_sum.
dataset activate Indicator.
sort cases by state.
match files
    /FILE=*
    /TABLE='DQState'
    /BY state
    /KEEP state to TwelveMoCohort DQFailedCheck_sum.
execute.

dataset close DQIndicator.
dataset close DQstate.

* Select only states that met the DQ checks.
select if DQFailedCheck_sum = 0.
execute.

delete variables DQFailedCheck_sum.

* Count the number of states remaining.
```

```
compute numstates = 1.
if (state eq lag(state))numstates = 0.
execute.

************************************************
SELECT APPLICABLE RECORDS
************************************************

* Select only the most recent 6-month record reported for each child, for
each episode.
************************************************

* This will create an episode-level file. Episodes are distinguished by
the date of latest removal from home.
* A child has a record in each 6-month submission until he discharged,
dropped, or was still in care as of
* the last reporting period we have in the file. We only want one record
per child, per episode, and the record we pick
* is the last one reported for that episode. This record will contain the
most recent data available that describes
* the child's episode (e.g., placement moves, age at entry, etc.).

* Flag the records to keep.
sort cases BY ChildID (A) DtLatRem (A).
match files
  /FILE=*
  /BY ChildID DtLatRem
  /LAST=Flag_LastRpt4Ep.
variable labels Flag_LastRpt4Ep 'last 6-month report we received for this
episode (based on DtLatRem)'.
value labels Flag_LastRpt4Ep 0 'Duplicate Case' 1 'Primary Case'.
variable level Flag_LastRpt4Ep (ORDINAL).
execute.

* Select the last 6-month record for each child, for each episode.
select if Flag_LastRpt4Ep=1.
execute.

delete variables Flag_LastRpt4Ep.

* Select only children who entered during the 12-month period specified
earlier.
************************************************

* This is the cohort of children for whom performance will be examined.

* Flag the records to keep.
if ((DtLatRem ge DtPeriodBeg) and (DtLatRem le DtPeriodEnd)) Entered = 1.
recode Entered (sysmis = 0).
execute.

variable labels Entered 'Child entered care during the specified 12-month
period'.
value labels Entered
```

```
0 'No'
1 'Yes'.

* Select the Entered records to keep.
select if Entered = 1.
execute.

* Handling multiple episodes in the 12-month period.
*****************************************************

* For placement stability, all of a child's episodes during the 12-month
period are considered.
* For example, if a child has two entries in the 12-month period, data
from both episodes are used.

*****************************************************
* REMOVE RECORDS WITH DQ PROBLEMS RELEVANT TO THIS MEASURE.
*****************************************************

* Do not delete dropped records if they occur in the last period needed
to calculate observed performance.
* This ensures we use only data from submissions required to observe the
cohort.
if DtReportEnd eq DtPeriodEnd DQ_Dropped = 0.
execute.

* Flag records with a problem (i.e., = 1) for any of the DQ checks used
for Placement Stability,
* except DQ_IDNoMatchNext6Mo and DQ_totalrem1.
compute DQ_Indicator=0.
if any(1,DQ_DOBgtDtDisch, DQ_DOBgtDtLatRem, DQ_Dropped,
DQ_DtDischeqDtLatRem, DQ_DtDischltDtLatRem, DQ_gt21DOBtoDtDisch,
DQ_gt21DOBtoDtLatRem, DQ_gt21DtDischtoDtLatRem, DQ_missDOB,
DQ_missDtLatRem, DQ_missNumPlep, DQ_dtpriordafterdlm)
DQ_Indicator=1.
execute.

* Delete records with a DQ problem(s).
select if DQ_Indicator=0.
execute.

* Delete the DQ variables as they are no longer needed.
delete variables DQ_Dropped to DQ_totalrem1 DQ_Indicator.


*****************************************************
CALCULATE LENGTH OF STAY FOR EACH EPISODE
*****************************************************

* Identify children who exited during the 12-month period (needed later).
*****************************************************
* Identify children who have more than one episode during the 12-month
period.
*****************************************************
```

```
sort cases by childid dtreportbeg.
compute duplicate_child eq 0.
execute.
if (childid eq lag(childid)) duplicate_child=1.
execute.

*label each episode.
compute position=1.
if (duplicate_child=1) position = lag(position)+1.
execute.

*flag all records for children with more than one episode.
aggregate
outfile=* mode=ADDVARIABLES
/break state childid
/dupchild = max(duplicate_child).

*pull dtpriordisch from into prior records.

PRESERVE.
set workspace 200000.
create TempDtPriorDisch=Lead(DtPriorDisch,1).
execute.
RESTORE.

*pull number of total removals reported in subsequent episode into record
from prior episode.
PRESERVE.
set workspace 200000.
create nexttotrem=Lead(totalrem,1).
execute.
RESTORE.

*if child has more than one episode, calculate the different between
total removals reported in second episode and total removals reported in
first episode.

if (dupchild=1 and position=1) changerem = (nexttotrem - totalrem).
execute.

*flag cases/records where we need to look at dtpriordisch.
compute flag_useprior = 0.
if not(missing(tempdtpriordisch)) AND missing(dtdisch) AND dupchild eq 1
AND position eq 1 flag_useprior = 1.
execute.

*flag when dtpriordisch falls in the period of analysis.
if flag_useprior eq 1 and tempdtpriordisch ge dtperiodbeg and
tempdtpriordisch le dtperiodend priorinperiod eq 1.
execute.

*flag if dtpriordisch falls in first 6-month period of the analysis.
```

```
if priorinperiod eq 1 and tempdtpriordisch le dtendfirst6mo priorinfirst
eq 1.
execute.

*flag when it is acceptable to use dtpriordisch to compute length of
stay.
if flag_useprior eq 1 and tempdtpriordisch ge dtperiodbeg and
tempdtpriordisch le DtEndFirst6Mo AND changerem eq 1 usepriordisch eq 1.
execute.

*drop episodes where to number of total removals changed by more than 1
between the first and second epsiode reported for the child.
compute drop_changerem =0.
if dupchild eq 1 and position eq 1 and changerem ne 1 and
missing(dtdisch) drop_changerem = 1.
execute.

select if drop_changerem ne 1.
execute.

*set DtDischNextRecord equal to tempdtpriordisch is child/episode meets
all requirements to use date prior discharge for computing LOS.
if  usepriordisch eq 1 AND adjust18 eq 1 and changerem eq 1 and
tempdtpriordisch gt dtlatrem AND (TempDtPriorDisch lt DtDischAdjusted)
AND (DtDischAdjusted eq Bday18) DtDischNextRecord = TempDtPriorDisch.
if usepriordisch eq 1 and missing(DtDischAdjusted) AND changerem eq 1 and
tempdtpriordisch gt dtlatrem DtDischNextRecord=TempDtPriorDisch.
execute.

*We can only use dtdischnextrecord to compute los if the date falls in
the first six month period.
* If the date falls in the second six month period we have to use the
last day of the first 6-month period to compute LOS.
if (DtDischNextRecord ge dtperiodbeg and DtDischNextRecord le
DtEndFirst6mo) DtDischAdjusted=DtDischNextRecord.
execute.

if ((DtDischAdjusted ge DtPeriodBeg) and (DtDischAdjusted le
DtPeriodEnd)) Exited = 1.
recode Exited (sysmis = 0).
execute.
variable labels Exited 'Child exited care during the specified 12-month
period'.
value labels Exited
0 'No'
1 'Yes'.
* LOS (days) between date of latest removal from home and date of
discharge (or last date in the 12-month period).
************************************************

compute LOSLatRemDays = $sysmis.
* For children who exited during the 12-month period, LOS is time between
DtLatRem and DtDischAdjusted.
do if Exited=1.
```

```
compute LOSLatRemDays=datediff(DtDischAdjusted,DtLatRem,"days").
end if.
execute.

* For children who did not exit during the 12-month period, LOS is time
between DtLatRem and DtPeriodEnd.
do if Exited=0.
compute LOSLatRemDays=datediff(DtPeriodEnd,DtLatRem,"days").
end if.
execute.
*For children who had more than one episode during the 12-month period
and the discharge date of the first episode fell in the second 6-month
period,
*LOS is the time between DtLatRem and the last day of the first 6-month
period.

do if Exited=0 AND dupchild=1 AND position=1 and (DtEndFirst6mo gt
DtLatRem).
compute LOSLatRemDays=datediff(DtEndFirst6mo,DtLatRem,"days").
end if.
execute.

*For children who had more than one episode during the 12-month period,
the LOS for the second episode if the child did not discharge from that
episode
*is the time between the removal date for that episode and the last day
of the 12-month period of analysis.
do if Exited=0 AND dupchild=1 AND position=2 and (DtPeriodEnd gt
DtLatRem).
compute LOSLatRemDays=datediff(DtPeriodEnd,DtLatRem,"days").
end if.
execute.

variable labels
LOSLatRemDays 'LOS - Days between DtLatRem and DtDischAdjusted (for
discharged) or last date in the 12-month period (for still in care)'.

formats LOSLatRemDays (F4.0).

************************************************
IDENTIFY AND REMOVE EPISODES THAT MEET EXCLUSION CRITERIA
************************************************

* The denominator for this indicator is the sum of children's LOS (days)
across *all* episodes during the
* 12-month period. So if a child entered twice during the 12-month
periods, his total LOS is the LOS from his
* first episode plus the LOS from his second episode. However, when
summing a child's LOS over all his episodes
* in the 12-month period, we want to exclude:
* 1) episodes in which the child entered at age 18 or more and
* 2) *complete* episodes with LOS < 8 days. If the episode has a LOS < 8
days, but the child is still in care, we
```

```
* want to keep this. These short but ongoing episodes represent entries
that occurred near the end of the
* 12-month period and continued past it (i.e., child was still in care).

* Complete episodes with LOS < 8 days *** THIS WILL CORRECTLY ELIMINATE
STAYS FOR CHILDREN WHO ENTERED WITHIN 8 DAYS OF 18th BDAY.
compute ExLOS8 = 0.
if (Exited = 1 and Adjust18 ne 1 and LOSLatRemDays < 8) ExLOS8 = 1.
execute.

variable labels
ExLOS8 'Complete episode has a LOS < 8 days'.

value labels ExLOS8
0 'No'
1 'Yes'.
execute.
formats ExLOS8 (F1.0).


* Select the records to keep.
select if (ExLOS8 = 0 and ExAge18 = 0).
execute.

*************************************************
ADJUST NUMBER OF PLACEMENTS
*************************************************

* We don't want to count the first placement (which reflects the child's
entry into care).

compute numplepAdjust = numplep -1.
if (DtCurSet gt DtDischAdjusted and numplepAdjust ge 1) numplepadjust =
numplepadjust - 1.
execute.

*formatting.
variable labels numplepAdjust 'Number of placement settings, excluding
the first placement setting associated with the childs entry'.

formats numplepAdjust (F2.0).

*flag records for same child with conflicting information.
sort cases by ChildID dtreportbeg.
if (dtlatrem ne lag(dtlatrem) AND childID eq lag(ChildID) and
missing(lag(dtdischadjusted))) flagCase eq 1.
execute.

*flag if child with two different dates of latest removal doesn't have a
dtpriordisch or the date is from before the period began.
if flagcase eq 1 AND (missing(dtpriordisch) or (dtpriordisch lt
dtperiodbeg)) flagcase2 eq 1.
execute.
```

```
**set flag to remove earliest record in the period for the child.

aggregate
  /OUTFILE=* MODE=ADDVARIABLES
  /BREAK=ChildID
  /Flagrptremove2 = max(flagcase2).

compute dropreport eq 0.
if (flagrptremove2 eq 1 AND position eq 1) dropreport eq 1.
execute.

select if dropreport eq 0.
execute.

*flag if child has two records that show overlapping episodes and remove
earliest report in period for child.
if (dtlatrem ne lag(dtlatrem)) AND childID eq lag(ChildID) AND
missing(dtpriordisch) AND dtlatrem lt lag(dtdischadjusted) flagpriorrec =
1.
execute.

aggregate
  /OUTFILE=* MODE=ADDVARIABLES
  /BREAK=ChildID
  /removerec = max(flagpriorrec).

compute removerecord = 0.
if (position eq 1 AND removerec eq 1) removerecord = 1.
execute.

select if removerecord eq 0.
execute.


*********************************************
SUM EACH CHILD's LOS and NUMBER OF PLACEMENTS ACROSS EPISODES
*************************************************

sort cases BY ChildID.
aggregate
  /OUTFILE=* MODE=ADDVARIABLES
  /BREAK=ChildID
  /Den_Child=SUM(LOSLatRemDays)
  /Num_Child=SUM(numplepAdjust).

variable labels
Den_Child 'Placement stability denominator - Childs total length of stay
(days) across all episodes in the 12-month period'
Num_Child 'Placement stability numerator - Childs total number of
placements (numplepAdjust) across all episodes in the 12-month period'.

formats Den_Child (F3.0) Num_Child (F2.0).
```

```
* Some children may have a total LOS = 0 days.
*************************************************

* These are children who entered on the last day of the 12-month period.
At this point, they have not been in care
* for a full 24 hours. In addition, placement rates for these children
cannot be calculated when their LOS
* (i.e., denominator) is 0.

select if Den_Child <> 0.
execute.

* Some children may have more moves than days in care, and therefore a
placement rate > 1.
*************************************************

* These are likely DQ issues. In the national file, affects only ~6
children. In addition, placement rates for these
* children cannot be used as rate could be > 1.

compute MovesgtDays = 0.
if numplepAdjust > Den_Child MovesgtDays = 1.
execute.

select if MovesgtDays = 0.
execute.

* Number of placement settings (categorical)
*************************************************
* Not used for CFSR 3 performance, but may be useful for reporting and
descriptive statistics.

recode Num_Child (1=1) (2=2) (3=3) (4=4) (5=5) (6=6) (7=7) (8=8) (9=9)
(10 thru Highest=10) (else=sysmis) into Num_ChildCat.
execute.

variable labels Num_ChildCat 'Total number of placement settings
(categorical)'.
value labels Num_ChildCat
1 '1'
2 '2'
3 '3'
4 '4'
5 '5'
6 '6'
7 '7'
8 '8'
9 '9'
10 '10 or more'.
formats Num_ChildCat (F2.0).
```

```
************************************************
AGGREGATE FILE TO ONE RECORD PER CHILD
************************************************

* For children with more than one episode, retain the age at entry and
totalrem associated with
* his earliest episode (based on DtLatRem) in the 12-month period.
************************************************

* Only age at entry is used in risk adjustment, but totalrem may be of
interest for reporting and
* descriptive statistics.
sort cases by ChildID(A) DtLatRem(A).
execute.
aggregate
  /OUTFILE=* MODE=ADDVARIABLES OVERWRITEVARS=YES
  /PRESORTED
  /BREAK=ChildID
  /TRemCat_first=FIRST(TRemCat)
  /AgeNmosyrsCat_first=FIRST(AgeNmosyrsCat)
  /AgeNmosyrs_first=FIRST(AgeNmosyrs).

* Select one record per child.
************************************************
* Select the most recently reported one for the 12-month period, although
the decision is arbitrary since
* the data used to measure performance is identical in both records
(i.e., Num_Child, Den_Child, Age...first).

* Flag the records to keep.
sort cases BY ChildID (A) DtReportEnd (A).
match files
  /FILE=*
  /BY ChildID
  /LAST=Flag_LastRpt4Ch.
variable labels Flag_LastRpt4Ch 'last 6-month report we received for this
child (based on DtReportEnd)'.
value labels Flag_LastRpt4Ch 0 'Duplicate Case' 1 'Primary Case'.
variable level Flag_LastRpt4Ch (ORDINAL).
execute.

* Select the records to keep.
select if Flag_LastRpt4Ch=1.
execute.

delete variables Flag_LastRpt4Ch.

************************************************
* SUMMARY STATISTICS
************************************************

* Create variables holding state and national performance.
************************************************
```

```
* Child-level data.
compute Perf_Child = (Num_Child / Den_Child).
compute Perf_Child_MP = (Num_Child / Den_Child) * 1000.
execute.

* State-level data.
sort cases by state (A) ChildID (A).
aggregate
  /OUTFILE=* MODE=ADDVARIABLES
  /PRESORTED
  /BREAK=state
  /Num_State=SUM(Num_Child)
  /Den_State=SUM(Den_Child)
  /N_State=N.
compute Perf_State = (Num_State / Den_State).
compute Perf_State_MP = (Num_State / Den_State) * 1000.
execute.

* National-level data.
aggregate
  /OUTFILE=* MODE=ADDVARIABLES
  /BREAK=
  /Num_Nation=SUM(Num_Child)
  /Den_Nation=SUM(Den_Child)
  /N_Nation=N.
compute Perf_Nation = (Num_Nation / Den_Nation).
compute Perf_Nation_MP = (Num_Nation / Den_Nation) * 1000.
execute.

formats Perf_Child Perf_State Perf_Nation (F6.5).

variable labels
N_State 'Number of children in the states file'
Den_State 'Placement stability denominator - Total number days children
were in care in state across all episides in the 12-month period'
Num_State 'Placement stability numerator - Total number of placement
moves in state children experienced while in care across all episodes in
the 12-month period'
Perf_State_MP 'Placement stability - Placement rate per 1,000 days in
care (state)'
N_Nation 'Number of children in the national file'
Den_Nation 'Placement stability denominator - Total number days children
were in care in nation across all episides in the 12-month period'
Num_Nation 'Placement stability numerator - Total number of placement
moves in nation children experienced while in care across all episodes in
the 12-month period'
Perf_Nation_MP 'Placement stability - Placement rate per 1,000 days in
care (nation)'.


************************************************
* OUTPUT FILES
************************************************
```

```
* Save.
save outfile='Observed_Child_FH\CFSR 3 - Observed perf for placement
stability ' + PeriodType + YYstr + '.sav'
   /compressed.

* Save file for STATA (for multi-level modeling).
rename variables (AgeNmosyrs_first = ChildAge).
save translate OUTFILE='Observed_Child_FH\CFSR 3 - Observed perf for
placement stability ' + PeriodType + YYstr + '.dta'
  /TYPE=STATA
  /VERSION=13
  /EDITION=SE
  /MAP
  /REPLACE
  /KEEP=state stateabb statetxt TwelveMoCohort DtPeriodBeg DtPeriodEnd
ChildID Num_Child Den_Child ChildAge N_State Den_State
Num_State Perf_State Perf_State_MP N_Nation Den_Nation Num_Nation
Perf_Nation Perf_Nation_MP.
rename variables (ChildAge = AgeNmosyrs_first).

* Save file with one record per state holding observed performance for
the 12-month cohort.
compute numstates = 1.
if (state eq lag(state))numstates = 0.
execute.

dataset copy OneRecordPerState.
dataset activate OneRecordPerState.
select if (numstates=1).
execute.
dataset activate OneRecordPerState.

string Indicator (A30).
execute.
compute Indicator = "Placement stability".
execute.

save outfile='Observed_State_FH\CFSR 3 - Observed perf for placement
stability State file ' + PeriodType + YYstr + '.sav'
   /keep state stateabb statetxt Indicator TwelveMoCohort DtPeriodBeg
DtPeriodEnd Perf_State.

dataset activate Indicator.
dataset close OneRecordPerState.

output save OUTFILE='SpssLogs_FH\CFSR 3 - Observed perf for placement
stability ' + PeriodType + YYstr + '.spv'.

************************************************************
ALL DONE.
************************************************************
```

CFSR 3: RISK-STANDARDIZED PERFORMANCE - MALTREATMENT IN FOSTER CARE

```
* VERSION HISTORY
*****************************

*****************************
*****************************
* CFSR 3: RISK-STANDARDIZED PERFORMANCE
* - MALTREATMENT IN FOSTER CARE
*****************************
*****************************

* This syntax is provided for informational purposes only. It requires
access to
* child-level data from all states, the District of Columbia, and Puerto
Rico.
* The following syntax is used to calculate an individual state's
performance
* for a recent cohort of children against the national standard and the
* historical cohorts (from all other states) that were used to establish
the
* national standard. Although this syntax will calculate RSPs and data
for all
* states, the only results of interest are for the state being evaluated
* (MyState "XX").

*****************************
* SPECIFY THE STATE AND 12-MONTH COHORT WHOSE PERFORMANCE IS BEING
ASSESSED
*****************************

* Note: For the 12-month cohort specified (CurrentCohort), there must
exist a
* file for that same cohort in "C:\cfsr3\Performance observed child".
This file
* will contain observed performance for a recent cohort of children (from
all
* states). For example, if the user specifies CurrentCohort = AB14, there
must
* be a file called "CFSR 3 - Observed perf for maltx in care AB14.dta."

set more off
set trace off
*set min_memory 16G
*****************************
* USER INPUT REQUIRED
*****************************

* Specify root folder where your files and folders are.
* This folder must have these subfolders:
* ... \Performance modeled
* ... \Performance observed child
* ... \Fixed files

local dirstub "C:\CFSR 3\Analysis - DP 2017 Aug"
```

```
* Specify 12-month cohort
local CurrentCohortList AB15
*local CurrentCohortList AB12 AB13 AB14 AB15 AB16

* Specify state(s) to run.
* Option 1 is to list them below in the local line (e.g.):
*local statelist CO MT
*local statetxtlist Colorado Montana
*      local statelist MA NH VT ME CT RI PA NY
* Option 2 is to list them in a dataset (which can be conveniently used
for the
* other indicators).
use "`dirstub'/1 - Fixed files/States 2017.dta", clear
local statelist
local statetxtlist
forvalues i = 1/`=_N' {
     local statelist `"`statelist' `"`=stateabb[`i']'"'"'
      local statetxtlist `"`statetxtlist' `"`=statetxt[`i']'"'"'
}
****************************
* RUN SYNTAX FROM HERE TO END
****************************
foreach CurrentCohort of local CurrentCohortList {
*local numelements = wordcount("`statelist'")
local numelements : word count `statelist'

*forvalue i = 1/`numelements'{
forvalue i = 1/1{
 local MyState : word `i' of `statelist'
 local MyStateTxt : word `i' of `statetxtlist'
* create log file.
log using "`dirstub'/4 - Performance modeled/RSP for maltx in care
`MyState' `CurrentCohort'.txt", text replace


****************************
* GET SOURCE FILE and PREP FILE
****************************

* Open the current cohort file. This file contains observed performance
for a
* recent cohort of children (from all states).
use "`dirstub'/3 - Performance observed child/CFSR 3 - Observed perf for
maltx in care `CurrentCohort'.dta", clear
keep if stateabb == `"`MyState'"'


* Flag if state requested is missing from the current cohort data file
(usually due to failing DQ).
quietly count
local DQState = 0
if r(N) == 0 {
      local DQState = 1
```

```
}

tempfile holding
save `holding'

* Open the historical cohort file. This file contains the fixed national
* standard for the indicator (Perf_Nation) and the observed performance
for the
* 12-month historical cohort of children that was used to establish the
national
* standard. We will be assessing the state's performance with its most
recent
* cohort (currently stored in 'holding') against the national standard
and
* historical cohorts from all other states. This syntax assumes the
historical
* cohort file is saved in the Fixes files folder of the dirstub path
defined earlier.
macro list
use "`dirstub'/1 - Fixed files/CFSR 3 - NS Observed perf for maltx in
care AB13.dta", clear

quietly summarize ChildAge, detail
local ChildAge_Med = r(p50)
local Perf_NationNS = Perf_Nation[1]
local Perf_NationNS_MP = Perf_Nation_MP[1]

* IF State requested does not have a current cohort score, creat a
special DQ
* output for that state rather than run the rest of the code.
if `DQState' == 1 {
      clear
      set obs 10
      generate stateabb = `"`MyState'"'
      generate state = `"`MyStateTxt'"'
      generate Indicator_new = "Maltreatment in care
(victimizations/100,000 days in care)"
      generate NS = string(round(`Perf_NationNS_MP',0.01))
      generate TwelveMoCohort = "FY" + "20" +
substr(`"`CurrentCohort'"',1,2)
      generate Period = substr(`"`CurrentCohort'"',3,2) + "AB,FY" +
substr(`"`CurrentCohort'"',3,2)
      generate DataUsed = substr(`"`CurrentCohort'"',3,2) + "AB,FY" +
substr(`"`CurrentCohort'"',3,2)
      generate Format = "Number"
      generate RSP_NS = ""
      generate RSP_Range = ""
      generate RSP_Range_Formatted = ""
      generate RSP_Pip = ""
      generate ValueType = "DataUsed2" in 1
      replace ValueType = "Den_State" in 2
      replace ValueType = "Num_State" in 3
      replace ValueType = "Perf_NationRnd" in 4
      replace ValueType = "Perf_StateRnd" in 5
```

```
      replace ValueType = "RSP_NS2" in 6
      replace ValueType = "RspLowRnd" in 7
      replace ValueType = "RSP_Range2" in 8
      replace ValueType = "RSPRnd" in 9
      replace ValueType = "RSPUppRnd" in 10
      generate Result = 0
      replace Result = round(`Perf_NationNS',0.001) in 4
      generate Result_String = "DQ"
      replace Result_String = DataUsed in 1
      replace Result_String = "" in 4
      replace Result_String = "" in 6
      generate MatchField = ""
      *Save file
      save "`dirstub'/4 - Performance modeled/CFSR 3 - RSP for maltx in
care - `CurrentCohort' `MyState'.dta", replace
      outsheet stateabb state     Indicator_new NS TwelveMoCohort Period
DataUsed Format RSP_NS RSP_Range RSP_Range_Formatted RSP_Pip ValueType
Result Result_String MatchField ///
      using "`dirstub'/6 - Data for Tableau/Stata
Output/RSP_MaltxInCare_Summary `CurrentCohort' `MyState'.csv", comma
nolabel replace
      export excel stateabb state  Indicator_new NS TwelveMoCohort Period
DataUsed Format RSP_NS RSP_Range RSP_Range_Formatted RSP_Pip ValueType
Result Result_String MatchField ///
      using "`dirstub'/6 - Data for Tableau/Stata
Output/RSP_MaltxInCare_Summary `CurrentCohort' `MyState'.xlsx", nolabel
replace firstrow (variables)
      *close log and return to the top of the loop
       log close
       continue
}

drop if stateabb == `"`MyState'"'
append using `holding'
* For the state being evaluated, replace the value it has for national
observed
* performance (which is based on the current cohort for all states) with
the
* fixed NS for this indicator. The level of precision matters for the
* calculations RSP, RSPUpp, and RSPLow matters.

* IMPORTANT - For Rate indicators (PS & MALTX FC), replace Perf_Nation_MP
as
* well. For the Rate indicators, the _MP version is used as the NS when
* comparing to the RSP.
replace Perf_Nation = `Perf_NationNS' if stateabb == `"`MyState'"'
replace Perf_Nation_MP = `Perf_NationNS_MP' if stateabb == `"`MyState'"'
sort state

* Describe dataset
* Pick one observation per state
* Count and verify number of states in file
* Calculate desired direction for performance on this indicator
* Verify desired 12-month cohort has been selected
```

```
describe
egen pickone = tag(stateabb)
count if pickone
gen Passing = "Below NS"
tab Perf_Nation_MP
tab Perf_Nation
tab TwelveMoCohort

****************************
* PREDICT OUTCOME
****************************

* Run multi-level poisson model predicting victimizations (Num_Child)
given
* one or more foster care episodes with a particular duration in days
* (Den_Child). Adjust for child age at entry or on first day (ChildAge).

********  POSSIBLE USER INPUT NEEDED ********

* The xtmepoisson command requires the user to enter the value of X in
* ib(X).ChildAge, where X represents the reference group for children in
the
* national dataset. The median age is used as the reference group. The ib
* function creates a dummy variable for each age value.
* To get the median age value run:

summarize ChildAge, detail


* The coded value to use is the one next to "50%." Enter that value in
ib(X)
* below.

timer on 1
xtmepoisson Num_Child ib(`ChildAge_Med').ChildAge, baselevels
exposure(Den_Child) || state:, variance
timer off 1
timer list 1
timer clear

predict xb, xb
predict re, reffects
predict rese, reses

* xb = natural logarithm of child's predicted number of victimizations
based on
* child's age and the number of days of the child's foster care
episode(s)
* (i.e., exposure to the event of interest), but without considering the
child's
* home state (i.e., ignoring the state's random effect)
```

```
* re = state's random effect (shift in the natural logarithm of the
child's
* predicted number of victimizations after considering the child's home
state;
* aka, Empirical Bayes intercept)

* rese - standard error of state's random effect

*****************************
* CALCULATE RISK-STANDARDIZED PERFORMANCE
*****************************

* 1. Calculate PREDICTED number of victimizations in each state
*****************************

* The predicted number of victimizations based on the state's performance
with
* its observed case mix. This is our best prediction of future
performance,
* assuming no change in case mix or policy. It is calculated as the sum
of each
* child's predicted number of victimizations, which is a complex function
of
* the child's value of xb (based on child's age and episode duration in
days)
* and his or her state's specific random effect).

sort state
gen double Child_Pred = exp(xb+re)
by state: egen double Num_Pred = total(Child_Pred)

* 2. Calculate EXPECTED number of victimizations in each state
*****************************

* The expected number of victimizations based on the nation's performance
with
* the state's case mix. This represents how many victimizations we would
expect
* for the state's children if they were treated in an "average" state. It
is
* calculated as the sum of each child's predicted number of
victimizations (xb),
* including the *average* intercept of all states (the average of the
* random effects across states is zero).

gen double Child_Exp = exp(xb)
by state: egen double Num_Exp = total(Child_Exp)

* 3. Calculate risk-standardized ratio and risk-standardized performance
*****************************

* Calculate ratio of predicted to expected
* Multiply ratio by national observed performance (i.e., the national
standard)
```

```
* to get RSP
gen double Ratio_PE = Num_Pred / Num_Exp
gen double RSP = (Num_Pred / Num_Exp) * Perf_Nation
gen double RSP_MP = RSP * 100000

* 4. Calculate 95% confidence intervals for RSP
*****************************

* Upper CI of the RSP
sort state
gen double UppNum = exp(xb+re+(1.96*rese))
by state: egen double UppNumSum = total(UppNum)
gen double UppDen = exp(xb)
by state: egen double UppDenSum = total(UppDen)
gen double RSPUpp = (UppNumSum/UppDenSum) * Perf_Nation
gen double RSPUpp_MP = RSPUpp * 100000

* Low CI of the RSP
gen double LowNum = exp(xb+re-(1.96*rese))
by state: egen double LowNumSum = total(LowNum)
gen double LowDen = exp(xb)
by state: egen double LowDenSum = total(LowDen)
gen double RSPLow = (LowNumSum/LowDenSum) * Perf_Nation
gen double RSPLow_MP = RSPLow * 100000

*****************************
* COMPARE STATE'S RSP TO NATIONAL STANDARD (NS)
*****************************

* When comparing the state's RSP (actually, the CIs) to the national
observed
* performance, use rounded versions. For rate indicators (maltreatment in
* foster care and placement stablility), use Perf_Nation_MP, RSRLow_MP,
etc.
* instead of the unmultiplied rate (e.g., Perf_Nation, RSRLow, which have
too
* many trailing 0s such that the rounded rates will all = 0; e.g.,
.00000631
* will get rounded to .00; whereas 6.3094187 will get rounded to 6.31).

* Round the national observed performance
clonevar Perf_NationRnd = Perf_Nation_MP
replace Perf_NationRnd = round(Perf_NationRnd,0.01)

* Round the CIs of the RSP
clonevar RSPLowRnd = RSPLow_MP
clonevar RSPUppRnd = RSPUpp_MP
replace RSPLowRnd = round(RSPLowRnd,0.01)
replace RSPUppRnd = round(RSPUppRnd,0.01)

* Compare CI's of the RSP relative to national observed performance
gen RSP_NS = "No diff"
replace RSP_NS = "Met" if RSPUppRnd < Perf_NationRnd
replace RSP_NS = "Not met" if RSPLowRnd > Perf_NationRnd
```

```
* Count number of states meeting, not meeting, and no different from NS
egen RSP_Met=total(RSP_NS=="Met" & pickone)
egen RSP_NotMet=total(RSP_NS=="Not met" & pickone)
egen RSP_NotDif=total(RSP_NS=="No diff" & pickone)

* Count number of states that must engage in a PIP
gen RSP_Pip = ""
replace RSP_Pip = "No PIP" if RSP_NS == "Met"
replace RSP_Pip = "No PIP" if RSP_NS == "No diff"
replace RSP_Pip = "PIP" if RSP_NS == "Not met"


* Create variable holding median age to include in output. The coded age
values
* are one higher than the actual age (e.g., a coded value of 7 represents
an
* actual age of 6), so the median age = coded value for median age *
minus 1.
sort state
by state: egen MedAge = median(ChildAge)
replace MedAge = MedAge -1

* Save
save "`dirstub'/4 - Performance modeled/CFSR 3 - RSP for maltx in care -
`CurrentCohort' `MyState'.dta", replace

****************************
* REPORTING
****************************

* Export to excel the state's results, formatted for profile

gen Indicator_new = "Maltreatment in care (victimizations/100,000 days in
care)"

* Round the RSP and Perf_State to 2 decimals (so it's, e.g., 3.57; avoids
having
* to format it later in Excel to show only 2 decimals)
gen RSPRnd = round(RSP_MP,0.01)
gen Perf_StateRnd = round(Perf_State_MP,0.01)

* gen RSP_new = round(RSP,0.001)*100 (e.g., "40.1")
gen RSP_Range = string(RSPLowRnd) + "-" + string(RSPUppRnd)
gen RSP_Range_Formatted = string(RSPLowRnd) + "-" + string(RSPUppRnd)
gen Format = "Number"
if substr(`"`CurrentCohort'"',1,2) ==  "AB"{
               gen Period = substr(`"`CurrentCohort'"',3,2) + "AB,FY" +
substr(`"`CurrentCohort'"',3,2)
}
else if substr(`"`CurrentCohort'"',1,2) == "BA" {
               gen Period = string(real(substr(`"`CurrentCohort'"',3,2))-
1) + "B" + substr(`"`CurrentCohort'"',3,2) + "A"
}
```

```
if substr(`"`CurrentCohort'"',1,2) ==  "AB"{
               gen DataUsed = substr(`"`CurrentCohort'"',3,2) + "A-" +
substr(`"`CurrentCohort'"',3,2) + "B, FY" +
substr(`"`CurrentCohort'"',3,2)
}
else if substr(`"`CurrentCohort'"',1,2) == "BA" {
               gen DataUsed =
string(real(substr(`"`CurrentCohort'"',3,2))-1) + "B-" +
string(real(substr(`"`CurrentCohort'"',3,2))+2) + "A"
}
*These variables only exist to ensure that there is both a column and
record with this data after reshaping below, and because tableau.
gen NS = string(round(`Perf_NationNS_MP',0.01))
gen RSP_Range2 = RSP_Range_Formatted
gen RSP_NS2 = RSP_NS
gen DataUsed2 = DataUsed
gen MatchField = ""

* RSP (e.g., .0035744702...)
* RSP_MP (e.g., 3.5744702)
* gen RSP_new = round(RSP_MP,0.01) (e.g., "3.57")
* gen RSPRange = string(RSPLowRnd) + " - " + string(RSPUppRnd)
* Perf_State_MP (e.g., 3.5348582)

* outsheet stateabb Indicator_new TwelveMoCohort RSP_new RSPRange
Perf_NationRnd RSP_NS RSP_Pip ///
* if (stateabb == `"`MyState'"' & pickone==1) using
"`dirstub'/Performance modeled/Profile_RSP_MaltxInCare_Summary
`CurrentCohort' `MyState'.csv", comma nolabel replace

outsheet stateabb statetxt Indicator_new NS TwelveMoCohort Period
DataUsed DataUsed2 Format RSP_NS RSP_NS2 RSP_Range RSP_Range_Formatted
RSP_Range2 RSP_Pip Den_State Num_State Perf_StateRnd RSPLowRnd RSPRnd
RSPUppRnd Perf_NationRnd MatchField ///
if (stateabb == `"`MyState'"' & pickone==1) using "`dirstub'/4 -
Performance modeled/RSP_MaltxInCare_Summary `CurrentCohort'
`MyState'.csv", comma nolabel replace


import delimited "`dirstub'/4 - Performance
modeled/RSP_MaltxInCare_Summary `CurrentCohort' `MyState'.csv", clear
case(preserve)

rename (Den_State Num_State Perf_StateRnd RSPLowRnd RSPRnd RSPUppRnd
Perf_NationRnd) Result=
rename (RSP_Range2 RSP_NS2 DataUsed2) str=

reshape long Result str, i(stateabb statetxt) j(ValueType) string
*reshape long result str, i(stateabb indicator_new ns twelvemocohort
period data_used format rsp_ns rsp_range rsp_range_formatted rsp_pip)
j(ValueType) string
replace Result = 0 if missing(Result)
```

```
* Use a string version of the State's name rather than a labeled numeric
version.
rename (str) Result_String
rename (statetxt) state


outsheet stateabbstate Indicator_new    NS    TwelveMoCohort   Period
      DataUsed    Format      RSP_NS      RSP_Range   RSP_Range_Formatted
      RSP_Pip     ValueType   Result      Result_String MatchField ///
 using "`dirstub'/6 - Data for Tableau/Stata
Output/RSP_MaltxInCare_Summary `CurrentCohort' `MyState'.csv", comma
nolabel replace

export excel stateabb state  Indicator_new NS TwelveMoCohort Period
DataUsed Format RSP_NS RSP_Range RSP_Range_Formatted RSP_Pip ValueType
Result Result_String MatchField ///
 using "`dirstub'/6 - Data for Tableau/Stata
Output/RSP_MaltxInCare_Summary `CurrentCohort' `MyState'.xlsx", nolabel
replace firstrow (variables)

*close log and return to the top of the loop
 log close
}
}
```

CFSR 3: RISK-STANDARDIZED PERFORMANCE - RECURRENCE OF MALTREATMENT

```
* VERSION HISTORY
*****************************

*****************************
*****************************
* CFSR 3: RISK-STANDARDIZED PERFORMANCE
* - RECURRENCE OF MALTREATMENT
*****************************
*****************************

* This syntax is provided for informational purposes only. It requires
access to
* child-level data from all states, the District of Columbia, and Puerto
Rico.

* The following syntax is used to calculate an individual state's
performance
* for a recent cohort of children against the national standard and the
* historical cohorts (from all other states) that were used to establish
the
* national standard. Although this syntax will calculate RSPs and data
for all
* states, the only results of interest are for the state being evaluated
* (MyState "XX").

*****************************
* SPECIFY THE STATE AND 12-MONTH COHORT WHOSE PERFORMANCE IS BEING
ASSESSED
*****************************

* Note: For the 12-month cohort specified (CurrentCohort), there must
exist a
* file for that same cohort in "C:\cfsr3\Performance observed child".
This file
* will contain observed performance for a recent cohort of children (from
all
* states). For example, if the user specifies CurrentCohort = 1314, their
must
* be a file called "CFSR 3 - Observed perf for maltx recurrence
1314.dta."

set more off
*set trace on
*set min_memory 16g

*****************************
* USER INPUT REQUIRED
*****************************

* Specify root folder where your files and folders are.
* This folder must have these subfolders:
* ... \Performance modeled
* ... \Performance observed child
* ... \Fixed files
```

```
local dirstub "C:\CFSR 3\Analysis - DP 2017 Aug"

* Specify 12-month cohort
local CurrentCohortList "1213"

* Specify state(s) to run.
* Option 1 is to list them below in the local line:
 local statelist PA NY
 local statetxtlist Pennsylvania "New York"

* Option 2 is to list them in a dataset (which can be conveniently used
for the other indicators).
*use "`dirstub'/1 - Fixed files/States 2017.dta", clear
*levelsof stateabb, local(statelist)
*levelsof statetxt, local(statetxtlist)
*local statelist
*local statetxtlist
*forvalues i = 1/`=_N' {
*     local statelist `"`statelist' `"`=stateabb[`i']'"'"'
*      local statetxtlist `"`statetxtlist' `"`=statetxt[`i']'"'"'
* }

****************************
* RUN SYNTAX FROM HERE TO END
****************************

foreach CurrentCohort of local CurrentCohortList {

local numelements : word count `statelist'
forvalue i = 1/`numelements'{
local MyState : word `i' of `statelist'
local MyStateTxt : word `i' of `statetxtlist'

* create log file.
log using "`dirstub'/4 - Performance modeled/RSP for maltx recurrence
`MyState' `CurrentCohort'.txt", text replace

****************************
* GET SOURCE FILE and PREP FILE
****************************

* Open the current cohort file. This file contains observed performance
for a
* recent cohort of children (from all states).
use "`dirstub'/3 - Performance observed child/CFSR 3 - Observed perf for
maltx recurrence `CurrentCohort'.dta", clear
keep if stateabb == `"`MyState'"'

* Flag if state requested is missing from the current cohort data file
(usually due to failing DQ).
quietly count
local DQState = 0
if r(N) == 0 {
```

```
      local DQState = 1
}

tempfile holding
save `holding'


* Open the historical cohort file. This file contains the fixed national
* standard for the indicator (Perf_Nation) and the observed performance
for the
* 12-month historical cohort of children that was used to establish the
national
* standard. We will be assessing the state's performance with its most
recent
* cohort (currently stored in 'holding') against the national standard
and
* historical cohorts from all other states. This syntax assumes the
historical
* cohort file is saved in the Fixes files folder of the dirstub path
defined earlier.
macro list
use "`dirstub'/1 - Fixed files/CFSR 3 - NS Observed perf for maltx
recurrence 1213.dta", clear

quietly summarize ChildAge, detail
local ChildAge_Med = r(p50)
local Perf_NationNS = Perf_Nation[1]

* IF State requested does not have a current cohort score, creat a
special DQ
* output for that state rather than run the rest of the code.
if `DQState' == 1 {
      clear
      set obs 10
      generate stateabb = `"`MyState'"'
      generate state = `"`MyStateTxt'"'
      generate Indicator_new = "Recurrence of maltreatment"
      generate NS = "" + string(`Perf_NationNS'*100,"%02.1f")+"%"
      generate TwelveMoCohort = "FY" + "20" +
substr(`"`CurrentCohort'"',1,2)
      generate Period = "FY" + substr(`"`CurrentCohort'"',1,2) + "-" +
substr(`"`CurrentCohort'"',3,2)
      generate DataUsed = Period
      generate Format = "Percent"
      generate RSP_NS = ""
      generate RSP_Range = ""
      generate RSP_Range_Formatted = ""
      generate RSP_Pip = ""
      generate ValueType = "DataUsed2" in 1
      replace ValueType = "Den_State" in 2
      replace ValueType = "Num_State" in 3
      replace ValueType = "Perf_NationRnd" in 4
      replace ValueType = "Perf_StateRnd" in 5
      replace ValueType = "RSP_NS2" in 6
```

```
      replace ValueType = "RspLowRnd" in 7
      replace ValueType = "RSP_Range2" in 8
      replace ValueType = "RSPRnd" in 9
      replace ValueType = "RSPUppRnd" in 10
      generate Result = 0
      replace Result = round(`Perf_NationNS',0.001) in 4
      generate Result_String = "DQ"
      replace Result_String = "FY" + substr(`"`CurrentCohort'"',1,2) + "
- " + substr(`"`CurrentCohort'"',3,2) in 1
      replace Result_String = "" in 4
      replace Result_String = "" in 6
      generate MatchField = ""
      *Save file
      save "`dirstub'/4 - Performance modeled/CFSR 3 - RSP for maltx
recurrence - `CurrentCohort' `MyState'.dta", replace
      outsheet stateabb state     Indicator_new NS TwelveMoCohort Period
DataUsed Format RSP_NS RSP_Range RSP_Range_Formatted RSP_Pip ValueType
Result Result_String MatchField ///
      using "`dirstub'/6 - Data for Tableau/Stata
Output/RSP_MalRecur_Summary `CurrentCohort' `MyState'.csv", comma nolabel
replace
      export excel stateabb state  Indicator_new NS TwelveMoCohort Period
DataUsed Format RSP_NS RSP_Range RSP_Range_Formatted RSP_Pip ValueType
Result Result_String MatchField ///
      using "`dirstub'/6 - Data for Tableau/Stata
Output/RSP_MalRecur_Summary `CurrentCohort' `MyState'.xlsx", nolabel
replace firstrow (variables)
      *close log and return to the top of the loop
       log close
       continue
}

drop if stateabb == `"`MyState'"'
append using `holding'

* For the state being evaluated, replace the value it has for national
observed
* performance (which is based on the current cohort for all states) with
the
* fixed NS for this indicator. The level of precision matters for the
* calculations RSP, RSPUpp, and RSPLow matters.
replace Perf_Nation = `Perf_NationNS' if stateabb == `"`MyState'"'

sort state

* Describe dataset
* Pick one observation per state
* Count and verify number of states in file
* Calculate desired direction for performance on this indicator
* Verify desired 12-month cohort has been selected

describe
egen pickone = tag(stateabb)
count if pickone
```

```
gen Passing = "Below NS"
tab Perf_Nation
tab TwelveMoCohort

*****************************
* PREDICT OUTCOME
*****************************

* Run multi-level logit model predicting maltreatment recurrence
(Num_Child = 1)
* Adjust for child age at initial victimization (ChildAge)
* The median age is used as the reference group.

timer on 1
xtmelogit Num_Child ib(`ChildAge_Med').ChildAge, baselevels || state:,
variance
timer off 1
timer list 1
timer clear

predict xb, xb
predict re, reffects
predict rese, reses

* xb = child's predicted log odds of recurrence based on child's age, but
*    without considering the child's home state (i.e., ignoring the
state's
*    random effect)
* re = state's random effect (shift in child's predicted log odds of
*    recurrence after considering the child's home state; aka, Empirical
*    Bayes intercept)
* rese - standard error of state's random effect

*****************************
* CALCULATE RISK-STANDARDIZED PERFORMANCE
*****************************

* 1. Calculate PREDICTED number of recurrences in each state
*****************************

* The predicted number of recurrences based on the state's performance
with
* its observed case mix. This is our best prediction of future
performance,
* assuming no change in case mix or policy. It is calculated as the sum
of each
* child's predicated probability of recurrence, which is a complex
function
* of the child's value of xb and his or her state's specific random
effect.

sort state
gen double Child_Pred = exp(xb+re)/(1+exp(xb+re))
by state: egen double Num_Pred = total(Child_Pred)
```

```
* 2. Calculate EXPECTED number of recurrences in each state
*****************************

* The expected number of recurrences based on the nation's performance
with
* the state's case mix. This represents how many recurrences we would
expect
* for the state's children if they were treated in an "average" state. It
is
* calculated as the sum of each child's predicted probability of
recurrence
* (xb), including the *average* intercept of all states (the average of
the
* random effects across states is zero).

gen double Child_Exp = exp(xb)/(1+exp(xb))
by state: egen double Num_Exp = total(Child_Exp)

* 3. Calculate risk-standardized ratio and risk-standardized performance
*****************************

* Calculate ratio of predicted to expected
* Multiply ratio by national observed performance (i.e., the national
standard)
* to get RSP
gen double Ratio_PE = Num_Pred / Num_Exp
gen double RSP = (Num_Pred / Num_Exp) * Perf_Nation

* 4. Calculate 95% confidence intervals for RSP
*****************************

* Upper CI
sort state
gen double UppNum = exp(xb+re+(1.96*rese))/(1+exp(xb+re+(1.96*rese)))
by state: egen double UppNumSum = total(UppNum)
gen double UppDen = exp(xb)/(1+exp(xb))
by state: egen double UppDenSum = total(UppDen)
gen double RSPUpp = (UppNumSum/UppDenSum) * Perf_Nation

* Lower CI
gen double LowNum = exp(xb+re-(1.96*rese))/(1+exp(xb+re-(1.96*rese)))
by state: egen double LowNumSum = total(LowNum)
gen double LowDen = exp(xb)/(1+exp(xb))
by state: egen double LowDenSum = total(LowDen)
gen double RSPLow = (LowNumSum/LowDenSum) * Perf_Nation

*****************************
* COMPARE STATE'S RSP TO NATIONAL STANDARD (NS)
*****************************

* When comparing state's RSP (actually, the CIs) to the national observed
* performance, use rounded versions.
```

```
* Round the national observed performance
clonevar Perf_NationRnd = Perf_Nation
replace Perf_NationRnd = round(Perf_NationRnd,0.001)

* Round the CIs of the RSP
clonevar RSPLowRnd = RSPLow
clonevar RSPUppRnd = RSPUpp
replace RSPLowRnd = round(RSPLowRnd,0.001)
replace RSPUppRnd = round(RSPUppRnd,0.001)

* Compare CI's of the RSP relative to national observed performance
gen RSP_NS = "No diff"
replace RSP_NS = "Met" if RSPUppRnd < Perf_NationRnd
replace RSP_NS = "Not met" if RSPLowRnd > Perf_NationRnd

* Count number of states meeting, not meeting, and no different from NS
egen RSP_Met=total(RSP_NS=="Met" & pickone)
egen RSP_NotMet=total(RSP_NS=="Not met" & pickone)
egen RSP_NotDif=total(RSP_NS=="No diff" & pickone)

* Count number of states that must engage in a PIP
gen RSP_Pip = ""
replace RSP_Pip = "No PIP" if RSP_NS == "Met"
replace RSP_Pip = "No PIP" if RSP_NS == "No diff"
replace RSP_Pip = "PIP" if RSP_NS == "Not met"


* Create variable holding median age at initial victimization to include
in
* output. The coded age values are one higher than the actual age (e.g.,
a coded
* value of 7 represents an actual age of 6), so median age = coded value
for
* median age minus 1.
sort state
by state: egen MedAge = median(ChildAge)
replace MedAge = MedAge -1

* Save
save "`dirstub'/4 - Performance modeled/CFSR 3 - RSP for maltx recurrence
- `CurrentCohort' `MyState'.dta", replace

****************************
* REPORTING
****************************

gen Indicator_new = "Recurrence of maltreatment"

* Round the RSP and Perf_State to 1 decimal (so it's, e.g., .421; avoids
having
* to format it later in Excel to show only 1 decimal)
clonevar RSPRnd = RSP
replace RSPRnd = round(RSPRnd,0.001)
clonevar Perf_StateRnd = Perf_State
```

```
replace Perf_StateRnd = round(Perf_StateRnd,0.001)

* gen RSP_new = round(RSP,0.001)*100 (e.g., "40.1")
gen RSP_Range = string(RSPLowRnd*100,"%02.1f") + "-" +
string(RSPUppRnd*100,"%02.1f")
gen RSP_Range_Formatted = string(RSPLowRnd*100,"%02.1f") + "%-" +
string(RSPUppRnd*100,"%02.1f") + "%"
gen Format = "Percent"
gen Period = "FY" + substr(`"`CurrentCohort'"',1,2) + "-" +
substr(`"`CurrentCohort'"',3,2)
gen DataUsed = "FY" + substr(`"`CurrentCohort'"',1,2) + "-" +
substr(`"`CurrentCohort'"',3,2)

* NOTE These variables only exist to ensure that for tableau there is
both a column and record with this data after reshaping below.
gen NS = string(`Perf_NationNS'*100,"%02.1f")+"%"
gen RSP_Range2 = RSP_Range_Formatted
gen RSP_NS2 = RSP_NS
gen DataUsed2 = DataUsed
gen MatchField = ""

outsheet stateabb statetxt Indicator_new NS TwelveMoCohort Period
DataUsed DataUsed2 Format RSP_NS RSP_NS2 RSP_Range RSP_Range_Formatted
RSP_Range2 RSP_Pip Den_State Num_State Perf_StateRnd RSPLowRnd RSPRnd
RSPUppRnd Perf_NationRnd MatchField ///
if (stateabb == `"`MyState'"' & pickone==1) using "`dirstub'/4 -
Performance modeled/RSP_MalRecur_Summary `CurrentCohort' `MyState'.csv",
comma nolabel replace


import delimited "`dirstub'/4 - Performance modeled/RSP_MalRecur_Summary
`CurrentCohort' `MyState'.csv", clear case(preserve)

rename (Den_State Num_State Perf_StateRnd RSPLowRnd RSPRnd RSPUppRnd
Perf_NationRnd) Result=
rename (RSP_Range2 RSP_NS2 DataUsed2) str=

reshape long Result str, i(stateabb statetxt) j(ValueType) string

replace Result = 0 if missing(Result)


rename (str) Result_String
rename (statetxt) state

outsheet stateabbstate Indicator_new    NS    TwelveMoCohort   Period
      DataUsed   Format     RSP_NS     RSP_Range   RSP_Range_Formatted
      RSP_Pip    ValueType  Result     Result_String MatchField ///
 using "`dirstub'/6 - Data for Tableau/Stata Output/RSP_MalRecur_Summary
`CurrentCohort' `MyState'.csv", comma nolabel replace

export excel stateabb state  Indicator_new NS TwelveMoCohort Period
DataUsed Format RSP_NS RSP_Range RSP_Range_Formatted RSP_Pip ValueType
Result Result_String MatchField ///
```

```
 using "`dirstub'/6 - Data for Tableau/Stata Output/RSP_MalRecur_Summary
`CurrentCohort' `MyState'.xlsx", nolabel replace firstrow (variables)

*close log and return to the top of the loop
 log close
}
}
```

CFSR 3: RISK-STANDARDIZED PERFORMANCE - PERMANENCY IN 12 MONTHS FOR CHILDREN ENTERING FOSTER CARE

```
* VERSION HISTORY
****************************

****************************
****************************
* CFSR 3: RISK-STANDARDIZED PERFORMANCE
* - PERMANENCY IN 12 MONTHS FOR CHILDREN ENTERING FOSTER CARE
****************************
****************************

* This syntax is provided for informational purposes only. It requires
access to
* child-level data from all states, the District of Columbia, and Puerto
Rico.

* The following syntax is used to calculate an individual state's
performance
* for a recent cohort of children against the national standard and the
* historical cohorts (from all other states) that were used to establish
the
* national standard. Although this syntax will calculate RSPs and data
for all
* states, the only results of interest are for the state being evaluated
* (MyState "XX").

****************************
* SPECIFY THE STATE AND 12-MONTH COHORT WHOSE PERFORMANCE IS BEING
ASSESSED
****************************

* Note: For the 12-month cohort specified (CurrentCohort), there must
exist a
* file for that same cohort in "[Root folder]\Performance observed
child".
* You specify the root folder below (local dirstub).
* This file will contain observed performance for a recent cohort of
children
* (from all states). For example, if the user specifies CurrentCohort =
AB12,
* their must be a file called "CFSR 3 - Observed perf for perm (entries)
AB12.dta."

set more off
set trace off
*set min_memory 16g

****************************
* USER INPUT REQUIRED
****************************

* Specify root folder where your files and folders are.
* This folder must have these subfolders:
* ... \Performance modeled
* ... \Performance observed child
```

```
* ... \Fixed files

local dirstub "C:\CFSR 3\Analysis - DP 2018 March"

* Specify 12-month cohort
 local CurrentCohortList  AB15
*local CurrentCohortList BA13 AB13 BA14 AB14 BA15 AB15 BA16 AB16 BA17

* Specify state(s) to run.
* Option 1 is to list them below in the local line (e.g.):

 local statelist MI MS
 local statetxtlist Michigan Mississippi
* Option 2 is to list them in a dataset (which can be conveniently used
for the
* other indicators).
*use "`dirstub'/1 - Fixed files/States 2017.dta", clear
*local statelist
*local statetxtlist
*forvalues i = 1/`=_N' {
     *local statelist `"`statelist' `"`=stateabb[`i']'"'"'
       *local statetxtlist `"`statetxtlist' `"`=statetxt[`i']'"'"'
 *}

****************************
* RUN SYNTAX FROM HERE TO END
****************************
macro list
foreach CurrentCohort of local CurrentCohortList {

*local numelements = wordcount("`statelist'")
local numelements : word count `statelist'
*forvalue i = 1/1{
forvalue i = 1/`numelements'{
 local MyState : word `i' of `statelist'
 local MyStateTxt : word `i' of `statetxtlist'
* create log file.
log using "`dirstub'/4 - Performance modeled/RSP for perm (entries)
`MyState' `CurrentCohort'.txt", text replace

****************************
* GET SOURCE FILE and PREP FILE
****************************

* Open the current cohort file. This file contains observed performance
for a
* recent cohort of children (from all states).
use "`dirstub'/3 - Performance observed child/CFSR 3 - Observed perf for
perm (entries) `CurrentCohort'.dta", clear
keep if stateabb == `"`MyState'"'


* Flag if state requested is missing from the current cohort data file
(usually due to failing DQ).
```

```
quietly count
local DQState = 0
if r(N) == 0 {
      local DQState = 1
}

tempfile holding
save `holding'

* Open the historical cohort file. This file contains the fixed national
* standard for the indicator (Perf_Nation) and the observed performance
for the
* 12-month historical cohort of children that was used to establish the
national
* standard. We will be assessing the state's performance with its most
recent
* cohort (currently stored in 'holding') against the national standard
and
* historical cohorts from all other states. This syntax assumes the
historical
* cohort file is saved in the Fixes files folder of the dirstub path
defined earlier.

use "`dirstub'/1 - Fixed files/CFSR 3 - NS Observed perf for perm
(entries) BA12.dta", clear


quietly summarize ChildAge, detail
local ChildAge_Med = r(p50)
local Perf_NationNS = Perf_Nation[1]

* IF State requested does not have a current cohort score, creat a
special DQ
* output for that state rather than run the rest of the code.
if `DQState' == 1 {
      clear
      set obs 10
      generate stateabb = `"`MyState'"'
      generate state = `"`MyStateTxt'"'
      generate Indicator_new = "Permanency in 12 months (entries)"
      generate NS = string(`Perf_NationNS'*100,"%02.1f")+"%"
      generate TwelveMoCohort = `"`CurrentCohort'"'
      if substr(`"`CurrentCohort'"',1,2) ==  "AB"{
                gen Period = substr(`"`CurrentCohort'"',3,2) + "A" +
substr(`"`CurrentCohort'"',3,2) + "B"
                      gen DataUsed = substr(`"`CurrentCohort'"',3,2) + "A-"
+ string(real(substr(`"`CurrentCohort'"',3,2))+2) + "B"
      }
      else if substr(`"`CurrentCohort'"',1,2) == "BA" {
                gen Period = string(real(substr(`"`CurrentCohort'"',3,2))-
1) + "B" + substr(`"`CurrentCohort'"',3,2) + "A"
                      gen DataUsed =
string(real(substr(`"`CurrentCohort'"',3,2))-1) + "B-" +
string(real(substr(`"`CurrentCohort'"',3,2))+2) + "A"
```

```
      }
      generate Format = "Percent"
      generate RSP_NS = ""
      generate RSP_Range = ""
      generate RSP_Range_Formatted = ""
      generate RSP_Pip = ""
      generate ValueType = "DataUsed2" in 1
      replace ValueType = "Den_State" in 2
      replace ValueType = "Num_State" in 3
      replace ValueType = "Perf_NationRnd" in 4
      replace ValueType = "Perf_StateRnd" in 5
      replace ValueType = "RSP_NS2" in 6
      replace ValueType = "RspLowRnd" in 7
      replace ValueType = "RSP_Range2" in 8
      replace ValueType = "RSPRnd" in 9
      replace ValueType = "RSPUppRnd" in 10
      generate Result = 0
      replace Result = round(`Perf_NationNS',0.001) in 4
      generate Result_String = "DQ"
      replace Result_String = DataUsed in 1
      replace Result_String = "" in 4
      replace Result_String = "" in 6
      generate MatchField = ""

      *Save file
      save "`dirstub'/4 - Performance modeled/CFSR 3 - RSP for perm
(entries) - `CurrentCohort' `MyState'.dta", replace
      outsheet stateabb state     Indicator_new NS TwelveMoCohort Period
DataUsed Format RSP_NS RSP_Range RSP_Range_Formatted RSP_Pip ValueType
Result Result_String MatchField ///
      using "`dirstub'/6 - Data for Tableau/Stata
Output/RSP_Perm12EN_Summary `CurrentCohort' `MyState'.csv", comma nolabel
replace
      export excel stateabb state  Indicator_new NS TwelveMoCohort Period
DataUsed Format RSP_NS RSP_Range RSP_Range_Formatted RSP_Pip ValueType
Result Result_String MatchField ///
      using "`dirstub'/6 - Data for Tableau/Stata
Output/RSP_Perm12FD24_more_Summary `CurrentCohort' `MyState'.xlsx",
nolabel replace firstrow (variables)
      *close log and return to the top of the loop
       log close
       continue
}


drop if stateabb == `"`MyState'"'
append using `holding'

* For the state being evaluated, replace the value it has for national
observed
* performance (which is based on the current cohort for all states) with
the
* fixed NS for this indicator. The level of precision matters for the
* calculations for RSP, RSPUpp, and RSPLow.
```

```
replace Perf_Nation = `Perf_NationNS' if stateabb == `"`MyState'"'

sort state

* Describe dataset
* Pick one observation per state
* Count and verify number of states in file
* Calculate desired direction for performance on this indicator
* Verify desired 12-month cohort has been selected

describe
egen pickone = tag(stateabb)
count if pickone
gen Passing = "Above NS"
tab Perf_Nation
tab TwelveMoCohort

****************************
* PREDICT OUTCOME
****************************

* Run multi-level logit model predicting permanency by 12 months
(Num_Child = 1)
* Adjust for child age at entry (ChildAge) and state's entry rate
(EntryRate)

********  POSSIBLE USER INPUT NEEDED ********

* The xtmelogit command requires the user to enter the value of X in
* ib(X).ChildAge, where X represents the reference group for children in
* the national dataset. The median age is used as the reference group.
The ib
* function creates a dummy variable for each age value.
* To get the median age value run:

summarize ChildAge, detail

* The coded value to use is the one next to "50%." Enter that value in
ib(X)
* below.

timer on 1
xtmelogit Num_Child ib(`ChildAge_Med').ChildAge EntryRate, baselevels ||
state:, variance
timer off 1
timer list 1
timer clear

predict xb, xb
predict re, reffects
predict rese, reses

* xb = child's predicted log odds of permanency based on child's age and
the
```

```
*    state-specific entry rate, but without considering the child's home
state
*    (i.e., ignoring the state's random effect)
* re = state's random effect (shift in child's predicted log odds of
*    permanency after considering the child's home state; aka, Empirical
*    Bayes intercept)
* rese - standard error of state's random effect

****************************
* CALCULATE RISK-STANDARDIZED PERFORMANCE
****************************

* 1. Calculate PREDICTED number of permanent exits in each state
****************************

* The predicted number of permanent exits based on the state's
performance with
* its observed case mix. This is our best prediction of future
performance,
* assuming no change in case mix or policy. It is calculated as the sum
of each
* child's predicated probability of permanent exit, which is a complex
function
* of the child's value of xb and his or her state's specific random
effect.

sort state
gen double Child_Pred = exp(xb+re)/(1+exp(xb+re))
by state: egen double Num_Pred = total(Child_Pred)

* 2. Calculate EXPECTED number of permanenct exits in each state
****************************

* The expected number of permanent exits based on the nation's
performance with
* the state's case mix. This represents how many permanent exits we would
expect
* for the state's children if they were treated in an "average" state. It
is
* calculated as the sum of each child's predicted probability of
permanent exit
* (xb), including the *average* intercept of all states (the average of
the
* random effects across states is zero).

gen double Child_Exp = exp(xb)/(1+exp(xb))
by state: egen double Num_Exp = total(Child_Exp)

* 3. Calculate risk-standardized ratio and risk-standardized performance
****************************

* Calculate ratio of predicted to expected
* Multiply ratio by national observed performance (i.e., the national
standard)
```

```
* to get RSP
gen double Ratio_PE = Num_Pred / Num_Exp
gen double RSP = (Num_Pred / Num_Exp) * Perf_Nation

* 4. Calculate 95% confidence intervals for RSP
*****************************

* Upper CI
sort state
gen double UppNum = exp(xb+re+(1.96*rese))/(1+exp(xb+re+(1.96*rese)))
by state: egen double UppNumSum = total(UppNum)
gen double UppDen = exp(xb)/(1+exp(xb))
by state: egen double UppDenSum = total(UppDen)
gen double RSPUpp = (UppNumSum/UppDenSum) * Perf_Nation

* Lower CI
gen double LowNum = exp(xb+re-(1.96*rese))/(1+exp(xb+re-(1.96*rese)))
by state: egen double LowNumSum = total(LowNum)
gen double LowDen = exp(xb)/(1+exp(xb))
by state: egen double LowDenSum = total(LowDen)
gen double RSPLow = (LowNumSum/LowDenSum) * Perf_Nation

*****************************
* COMPARE STATE'S RSP TO NATIONAL STANDARD (NS)
*****************************

* When comparing the state's RSP (actually, the CIs) to the national
observed
* performance, use rounded versions.

* Round the national observed performance
clonevar Perf_NationRnd = Perf_Nation
replace Perf_NationRnd = round(Perf_NationRnd,0.001)

* Round the CIs of the RSP
clonevar RSPLowRnd = RSPLow
clonevar RSPUppRnd = RSPUpp
replace RSPLowRnd = round(RSPLowRnd,0.001)
replace RSPUppRnd = round(RSPUppRnd,0.001)

* Compare CI's of the RSP relative to national observed performance
gen RSP_NS = "No diff"
replace RSP_NS = "Met" if RSPLowRnd > Perf_NationRnd
replace RSP_NS = "Not met" if RSPUppRnd < Perf_NationRnd

* Count number of states meeting, not meeting, and no different from NS
egen RSP_Met=total(RSP_NS=="Met" & pickone)
egen RSP_NotMet=total(RSP_NS=="Not met" & pickone)
egen RSP_NotDif=total(RSP_NS=="No diff" & pickone)

* Count number of states that must engage in a PIP
gen RSP_Pip = ""
replace RSP_Pip = "No PIP" if RSP_NS == "Met"
replace RSP_Pip = "No PIP" if RSP_NS == "No diff"
```

```
replace RSP_Pip = "PIP" if RSP_NS == "Not met"


* Create variable holding median age at entry to include in output. The
coded
* age values are one higher than the actual age (e.g., a coded value of 7
* represents an actual age of 6), so median age = coded value for median
age
* minus 1.
sort state
by state: egen MedAge = median(ChildAge)
replace MedAge = MedAge -1

* Save
save "`dirstub'/4 - Performance modeled/CFSR 3 - RSP for perm (entries) -
`CurrentCohort' `MyState'.dta", replace

****************************
* REPORTING
****************************

* Export to excel the state's results, formatted for profile

gen Indicator_new = "Permanency in 12 months (entries)"
gen RSPRnd = round(RSP,0.001)
gen Perf_StateRnd = round(Perf_State,0.001)

gen RSP_Range = string(RSPLowRnd*100,"%02.1f") + "-" +
string(RSPUppRnd*100,"%02.1f")
gen RSP_Range_Formatted = string(RSPLowRnd*100,"%02.1f") + "%-" +
string(RSPUppRnd*100,"%02.1f") + "%"

gen Format = "Percent"
if substr(`"`CurrentCohort'"',1,2) ==  "AB"{
               gen Period = substr(`"`CurrentCohort'"',3,2) + "A" +
substr(`"`CurrentCohort'"',3,2) + "B"
                    gen DataUsed = substr(`"`CurrentCohort'"',3,2) + "A-"
+ string(real(substr(`"`CurrentCohort'"',3,2))+2) + "B"
}
else if substr(`"`CurrentCohort'"',1,2) == "BA" {
               gen Period = string(real(substr(`"`CurrentCohort'"',3,2))-
1) + "B" + substr(`"`CurrentCohort'"',3,2) + "A"
                    gen DataUsed =
string(real(substr(`"`CurrentCohort'"',3,2))-1) + "B-" +
string(real(substr(`"`CurrentCohort'"',3,2))+2) + "A"
}

* MH These variables only exist to ensure that there is both a column and
record with this data after reshaping below, and because tableau.
gen NS = string(`Perf_NationNS'*100,"%02.1f")+"%"
gen RSP_Range2 = RSP_Range_Formatted
gen RSP_NS2 = RSP_NS
gen DataUsed2 = DataUsed
gen MatchField = ""
```

```
* gen Perf_NationRnd_new = Perf_NationRnd*100

* outsheet stateabb Indicator_new TwelveMoCohort RSP_new RSPRange
Perf_NationRnd_new RSP_NS RSP_Pip ///
* if (stateabb == `"`MyState'"' & pickone==1) using
"Profile_RSP_Perm12EN_Summary `CurrentCohort' `MyState'.csv", comma
nolabel replace

outsheet stateabb statetxt Indicator_new NS TwelveMoCohort Period
DataUsed DataUsed2 Format RSP_NS RSP_NS2 RSP_Range RSP_Range_Formatted
RSP_Range2 RSP_Pip Den_State Num_State Perf_StateRnd RSPLowRnd RSPRnd
RSPUppRnd Perf_NationRnd MatchField ///
if (stateabb == `"`MyState'"' & pickone==1) using "`dirstub'/4 -
Performance modeled/RSP_Perm12EN_Summary `CurrentCohort' `MyState'.csv",
comma nolabel replace


import delimited "`dirstub'/4 - Performance modeled/RSP_Perm12EN_Summary
`CurrentCohort' `MyState'.csv", clear case(preserve)

rename (Den_State Num_State Perf_StateRnd RSPLowRnd RSPRnd RSPUppRnd
Perf_NationRnd) Result=
rename (RSP_Range2 RSP_NS2 DataUsed2) str=

reshape long Result str, i(stateabb statetxt) j(ValueType) string
*reshape long result str, i(stateabb statetxt indicator_new ns
twelvemocohort period data_used format rsp_ns rsp_range
rsp_range_formatted rsp_pip) j(ValueType) string
replace Result = 0 if missing(Result)


rename (str) Result_String
rename (statetxt) state

outsheet stateabbstate Indicator_new    NS    TwelveMoCohort   Period
      DataUsed    Format      RSP_NS      RSP_Range   RSP_Range_Formatted
      RSP_Pip     ValueType   Result      Result_String MatchField ///
 using "`dirstub'/6 - Data for Tableau/Stata Output/RSP_Perm12EN_Summary
`CurrentCohort' `MyState'.csv", comma nolabel replace
 log close
}
}
```

CFSR 3: RISK-STANDARDIZED PERFORMANCE - PERMANENCY IN 12 MONTHS FOR CHILDREN IN CARE ON FIRST DAY 12-23 MONTHS

```
* VERSION HISTORY
****************************

****************************
****************************
* CFSR 3: RISK-STANDARDIZED PERFORMANCE
* - PERMANENCY IN 12 MONTHS FOR CHILDREN IN CARE ON FIRST DAY 12-23
MONTHS
****************************
****************************

* This syntax is provided for informational purposes only. It requires
access to
* child-level data from all states, the District of Columbia, and Puerto
Rico.
* The following syntax is used to calculate an individual state's
performance
* for a recent cohort of children against the national standard and the
* historical cohorts (from all other states) that were used to establish
the
* national standard. Although this syntax will calculate RSPs and data
for all
* states, the only results of interest are for the state being evaluated
* (MyState "XX").

****************************
* SPECIFY THE STATE AND 12-MONTH COHORT WHOSE PERFORMANCE IS BEING
ASSESSED
****************************

* Note: For the 12-month cohort specified (CurrentCohort), there must
exist a
* file for that same cohort in "C:\cfsr3\Performance observed child".
This file
* will contain observed performance for a recent cohort of children (from
all
* states). For example, if the user specifies CurrentCohort = AB14, their
must
* be a file called "CFSR 3 - Observed perf for perm (FD 12-23) AB14.dta."

set more off
set trace off
*set min_memory 16g

****************************
* USER INPUT REQUIRED
****************************

* Specify root folder where your files and folders are.
* This folder must have these subfolders:
* ... \Performance modeled
* ... \Performance observed child
* ... \Fixed files
```

```
local dirstub "C:\CFSR 3\Analysis - DP 2018 March"


* Specify 12-month cohort
local CurrentCohortList AB17
*local CurrentCohortList BA13 AB13 BA14 AB14 BA15 AB15 BA16 AB16 BA17


* Specify state(s) to run.
* Option 1 is to list them below in the local line (e.g.):
local statelist MI MS
local statetxtlist Michigan Mississippi
*      local statelist MA NH VT ME CT RI PA NY
* Option 2 is to list them in a dataset (which can be conveniently used
for the
* other indicators).
*use "`dirstub'/1 - Fixed files/States 2017.dta", clear
*local statelist
*local statetxtlist
*forvalues i = 1/`=_N' {
     *local statelist `"`statelist' `"`=stateabb[`i']'"'"'
      *local statetxtlist `"`statetxtlist' `"`=statetxt[`i']'"'"'

*****************************
* RUN SYNTAX FROM HERE TO END
*****************************
foreach CurrentCohort of local CurrentCohortList {

local numelements : word count `statelist'
forvalue i = 1/`numelements'{
local MyState : word `i' of `statelist'
local MyStateTxt : word `i' of `statetxtlist'
* create log file.
log using "`dirstub'/4 - Performance modeled/RSP for perm (FD 12-23)
`MyState' `CurrentCohort'.txt", text replace

*****************************
* GET SOURCE FILE and PREP FILE
*****************************

* Open the current cohort file. This file contains observed performance
for a
* recent cohort of children (from all states).
use "`dirstub'/3 - Performance observed child/CFSR 3 - Observed perf for
perm (FD 12-23) `CurrentCohort'.dta", clear

keep if stateabb == `"`MyState'"'
* tempfile holding
* save `holding'

quietly count
local DQState = 0
if r(N) == 0 {
      local DQState = 1
```

```
}

tempfile holding
save `holding'

* Open the historical cohort file. This file contains the fixed national
* standard for the indicator (Perf_Nation) and the observed performance
for the
* 12-month historical cohort of children that was used to establish the
national
* standard. We will be assessing the state's performance with its most
recent
* cohort (currently stored in 'holding') against the national standard
and
* historical cohorts from all other states. This syntax assumes the
historical
* cohort file is saved in the Fixes files folder of the dirstub path
defined earlier.

use "`dirstub'/1 - Fixed files/CFSR 3 - NS Observed perf for perm (FD 12-
23) BA14.dta", clear

quietly summarize ChildAge, detail
local ChildAge_Med = r(p50)
local Perf_NationNS = Perf_Nation[1]


* IF State requested does not have a current cohort score, creat a
special DQ
* output for that state rather than run the rest of the code.
if `DQState' == 1 {
      clear
      set obs 10
      generate stateabb = `"`MyState'"'
      generate state = `"`MyStateTxt'"'
      generate Indicator_new = "Permanency in 12 months (12 - 23 mos)"
      generate NS = string(`Perf_NationNS'*100,"%02.1f")+"%"
      generate TwelveMoCohort = `"`CurrentCohort'"'
      if substr(`"`CurrentCohort'"',1,2) ==  "AB"{
            gen Period = substr(`"`CurrentCohort'"',3,2) + "A" +
substr(`"`CurrentCohort'"',3,2) + "B"
            gen DataUsed = substr(`"`CurrentCohort'"',3,2) + "A-" +
substr(`"`CurrentCohort'"',3,2) + "B"
      }
      else if substr(`"`CurrentCohort'"',1,2) == "BA" {
            gen Period = string(real(substr(`"`CurrentCohort'"',3,2))-1)
+ "B" + substr(`"`CurrentCohort'"',3,2) + "A"
            gen DataUsed = string(real(substr(`"`CurrentCohort'"',3,2))-
1) + "B-" + substr(`"`CurrentCohort'"',3,2) + "A"
      }
      generate Format = "Percent"
      generate RSP_NS = ""
      generate RSP_Range = ""
      generate RSP_Range_Formatted = ""
```

```
      generate RSP_Pip = ""
      generate ValueType = "DataUsed2" in 1
      replace ValueType = "Den_State" in 2
      replace ValueType = "Num_State" in 3
      replace ValueType = "Perf_NationRnd" in 4
      replace ValueType = "Perf_StateRnd" in 5
      replace ValueType = "RSP_NS2" in 6
      replace ValueType = "RspLowRnd" in 7
      replace ValueType = "RSP_Range2" in 8
      replace ValueType = "RSPRnd" in 9
      replace ValueType = "RSPUppRnd" in 10
      generate Result = 0
      replace Result = round(`Perf_NationNS',0.001) in 4
      generate Result_String = "DQ"
      replace Result_String = DataUsed in 1
      replace Result_String = "" in 4
      replace Result_String = "" in 6
      generate MatchField = ""
      *Save file
      save "`dirstub'/4 - Performance modeled/CFSR 3 - RSP for perm (FD
12-23) - `CurrentCohort' `MyState'.dta", replace
      outsheet stateabb state     Indicator_new NS TwelveMoCohort Period
DataUsed Format RSP_NS RSP_Range RSP_Range_Formatted RSP_Pip ValueType
Result Result_String MatchField ///
      using "`dirstub'/6 - Data for Tableau/Stata
Output/RSP_Perm12FD12_23_Summary `CurrentCohort' `MyState'.csv", comma
nolabel replace

      *close log and return to the top of the loop
       log close
       continue
}

drop if stateabb == `"`MyState'"'
append using `holding'

* For the state being evaluated, replace the value it has for national
observed
* performance (which is based on the current cohort for all states) with
the
* fixed NS for this indicator. The level of precision matters for the
* calculations RSP, RSPUpp, and RSPLow matters.
replace Perf_Nation = `Perf_NationNS' if stateabb == `"`MyState'"'

sort state

* Describe dataset
* Pick one observation per state
* Count and verify number of states in file
* Calculate desired direction for performance on this indicator
* Verify desired 12-month cohort has been selected

describe
egen pickone = tag(stateabb)
```

```
count if pickone
gen Passing = "Above NS"
tab Perf_Nation
tab TwelveMoCohort

*****************************
* PREDICT OUTCOME
*****************************

* Run multi-level logit model predicting permanency by 12 months
(Num_Child = 1)
* Adjust for child age on first day (ChildAge)

********  USER INPUT POSSIBLY NEEDED ********

* The xtmelogit command requires the user to enter the value of X in
* ib(X).ChildAge, where X represents the reference group for children in
* the national dataset. The median age is used as the reference group.
The ib
* function creates a dummy variable for each age value.
* To get the median age value run:

summarize ChildAge, detail

* The coded value to use is the one next to "50%." Enter that value in
ib(X)
* below.

timer on 1
xtmelogit Num_Child ib(`ChildAge_Med').ChildAge, baselevels || state:,
variance
timer off 1
timer list 1
timer clear

predict xb, xb
predict re, reffects
predict rese, reses

* xb = child's predicted log odds of permanency based on child's age, but
*      without considering the child's home state (i.e., ignoring the
state's
*    random effect)
* re = state's random effect (shift in child's predicted log odds of
*    permanency after considering the child's home state; aka, Empirical
*    Bayes intercept)
* rese - standard error of state's random effect

*****************************
* CALCULATE RISK-STANDARDIZED PERFORMANCE
*****************************

* 1. Calculate PREDICTED number of permanent exits in each state
*****************************
```

```
* The predicted number of permanent exits based on the state's
performance with
* its observed case mix. This is our best prediction of future
performance,
* assuming no change in case mix or policy. It is calculated as the sum
of each
* child's predicated probability of permanent exit, which is a complex
function
* of the child's value of xb and his or her state's specific random
effect.

sort state
gen double Child_Pred = exp(xb+re)/(1+exp(xb+re))
by state: egen double Num_Pred = total(Child_Pred)

* 2. Calculate EXPECTED number of permanenct exits in each state
*****************************

* The expected number of permanent exits based on the nation's
performance with
* the state's case mix. This represents how many permanent exits we would
expect
* for the state's children if they were treated in an "average" state. It
is
* calculated as the sum of each child's predicted probability of
permanent exit
* (xb), including the *average* intercept of all states (the average of
the
* random effects across states is zero).

gen double Child_Exp = exp(xb)/(1+exp(xb))
by state: egen double Num_Exp = total(Child_Exp)

* 3. Calculate risk-standardized ratio and risk-standardized performance
*****************************

* Calculate ratio of predicted to expected
* Multiply ratio by national observed performance (i.e., the national
standard)
* to get RSP
gen double Ratio_PE = Num_Pred / Num_Exp
gen double RSP = (Num_Pred / Num_Exp) * Perf_Nation

* 4. Calculate 95% confidence intervals for RSP
*****************************

* Upper CI
sort state
gen double UppNum = exp(xb+re+(1.96*rese))/(1+exp(xb+re+(1.96*rese)))
by state: egen double UppNumSum = total(UppNum)
gen double UppDen = exp(xb)/(1+exp(xb))
by state: egen double UppDenSum = total(UppDen)
gen double RSPUpp = (UppNumSum/UppDenSum) * Perf_Nation
```

```
* Lower CI
gen double LowNum = exp(xb+re-(1.96*rese))/(1+exp(xb+re-(1.96*rese)))
by state: egen double LowNumSum = total(LowNum)
gen double LowDen = exp(xb)/(1+exp(xb))
by state: egen double LowDenSum = total(LowDen)
gen double RSPLow = (LowNumSum/LowDenSum) * Perf_Nation

*****************************
* COMPARE STATE'S RSP TO NATIONAL STANDARD (NS)
*****************************

* When comparing the state's RSP (actually, the CIs) to the national
observed
* performance, use rounded versions.

* Round the national observed performance
clonevar Perf_NationRnd = Perf_Nation
replace Perf_NationRnd = round(Perf_NationRnd,0.001)

* Round the CIs of the RSP
clonevar RSPLowRnd = RSPLow
clonevar RSPUppRnd = RSPUpp
replace RSPLowRnd = round(RSPLowRnd,0.001)
replace RSPUppRnd = round(RSPUppRnd,0.001)

* Compare CI's of the RSP relative to national observed performance
gen RSP_NS = "No diff"
replace RSP_NS = "Met" if RSPLowRnd > Perf_NationRnd
replace RSP_NS = "Not met" if RSPUppRnd < Perf_NationRnd

* Count number of states meeting, not meeting, and no different from NS
egen RSP_Met=total(RSP_NS=="Met" & pickone)
egen RSP_NotMet=total(RSP_NS=="Not met" & pickone)
egen RSP_NotDif=total(RSP_NS=="No diff" & pickone)

* Count number of states that must engage in a PIP
gen RSP_Pip = ""
replace RSP_Pip = "No PIP" if RSP_NS == "Met"
replace RSP_Pip = "No PIP" if RSP_NS == "No diff"
replace RSP_Pip = "PIP" if RSP_NS == "Not met"


* Create variable holding median age on first day to include in output.
The
* coded age values are identical to the actual age (e.g., a coded value
of 6
* represents an actual age of 6).
sort state
by state: egen MedAge = median(ChildAge)

* Save
save "`dirstub'/4 - Performance modeled/CFSR 3 - RSP for perm (FD 12-23)
- `CurrentCohort' `MyState'.dta", replace
```

```
***************************
* REPORTING
***************************

* Export to excel the state's results, formatted for profile

gen Indicator_new = "Permanency in 12 months (12 - 23 mos)"

* Round the RSP and Perf_State to 1 decimal (so it's, e.g., .421; avoids
having
* to format it later in Excel to show only 1 decimal)
clonevar RSPRnd = RSP
replace RSPRnd = round(RSPRnd,0.001)
clonevar Perf_StateRnd = Perf_State
replace Perf_StateRnd = round(Perf_StateRnd,0.001)

* gen RSP_new = round(RSP,0.001)*100 (e.g., "40.1")
gen RSP_Range = string(RSPLowRnd*100,"%02.1f") + "-" +
string(RSPUppRnd*100,"%02.1f")
gen RSP_Range_Formatted = string(RSPLowRnd*100,"%02.1f") + "%-" +
string(RSPUppRnd*100,"%02.1f") + "%"
gen Format = "Percent"
*gen Period = "FY" + substr(`"`CurrentCohort'"',1,2) + "-" +
substr(`"`CurrentCohort'"',3,2)
*gen DataUsed = "FY" + substr(`"`CurrentCohort'"',1,2) + "-" +
substr(`"`CurrentCohort'"',3,2)
if substr(`"`CurrentCohort'"',1,2) ==  "AB"{
      gen Period = substr(`"`CurrentCohort'"',3,2) + "A" +
substr(`"`CurrentCohort'"',3,2) + "B"
      gen DataUsed = substr(`"`CurrentCohort'"',3,2) + "A-" +
substr(`"`CurrentCohort'"',3,2) + "B"
}
else if substr(`"`CurrentCohort'"',1,2) == "BA" {
      gen Period = string(real(substr(`"`CurrentCohort'"',3,2))-1) + "B"
+ substr(`"`CurrentCohort'"',3,2) + "A"
      gen DataUsed = string(real(substr(`"`CurrentCohort'"',3,2))-1) +
"B-" + substr(`"`CurrentCohort'"',3,2) + "A"
}


* MH These variables only exist to ensure that there is both a column and
record with this data after reshaping below, and because tableau.
gen NS = string(`Perf_NationNS'*100,"%02.1f")+"%"
gen RSP_Range2 = RSP_Range_Formatted
gen RSP_NS2 = RSP_NS
gen DataUsed2 = DataUsed
generate MatchField = ""
* gen Perf_NationRnd_new = Perf_NationRnd*100

* outsheet stateabb Indicator_new TwelveMoCohort RSP_new RSPRange
Perf_NationRnd_new RSP_NS RSP_Pip ///
* if (stateabb == `"`MyState'"' & pickone==1) using
"Profile_RSP_Perm12EN_Summary `CurrentCohort' `MyState'.csv", comma
nolabel replace
```

```
outsheet stateabb statetxt Indicator_new NS TwelveMoCohort Period
DataUsed DataUsed2 Format RSP_NS RSP_NS2 RSP_Range RSP_Range_Formatted
RSP_Range2 RSP_Pip Den_State Num_State Perf_StateRnd RSPLowRnd RSPRnd
RSPUppRnd Perf_NationRnd MatchField ///
if (stateabb == `"`MyState'"' & pickone==1) using "`dirstub'/4 -
Performance modeled/RSP_Perm12FD12_23_Summary `CurrentCohort'
`MyState'.csv", comma nolabel replace



import delimited "`dirstub'/4 - Performance
modeled/RSP_Perm12FD12_23_Summary `CurrentCohort' `MyState'.csv", clear
case(preserve)

rename (Den_State Num_State Perf_StateRnd RSPLowRnd RSPRnd RSPUppRnd
Perf_NationRnd) Result=
rename (RSP_Range2 RSP_NS2 DataUsed2) str=

reshape long Result str, i(stateabb statetxt) j(ValueType) string
*reshape long result str, i(stateabb statetxt indicator_new ns
twelvemocohort period data_used format rsp_ns rsp_range
rsp_range_formatted rsp_pip) j(ValueType) string
replace Result = 0 if missing(Result)


rename (str) Result_String
rename (statetxt) state

outsheet stateabbstate Indicator_new    NS    TwelveMoCohort   Period
      DataUsed    Format      RSP_NS      RSP_Range   RSP_Range_Formatted
      RSP_Pip     ValueType   Result      Result_String MatchField ///
 using "`dirstub'/6 - Data for Tableau/Stata
Output/RSP_Perm12FD12_23_Summary `CurrentCohort' `MyState'.csv", comma
nolabel replace


 export excel stateabb state Indicator_new NS TwelveMoCohort Period
DataUsed Format RSP_NS RSP_Range RSP_Range_Formatted RSP_Pip ValueType
Result Result_String MatchField ///
 using "`dirstub'/6 - Data for Tableau/Stata
Output/RSP_Perm12FD12_23_Summary `CurrentCohort' `MyState'.xlsx", nolabel
replace firstrow (variables)
 log close

}
}
```

CFSR 3: RISK-STANDARDIZED PERFORMANCE - PERMANENCY IN 12 MONTHS FOR CHILDREN IN CARE ON FIRST DAY 24 MONTHS OR MORE

```
* VERSION HISTORY
*****************************

*****************************
*****************************
* CFSR 3: RISK-STANDARDIZED PERFORMANCE
* - PERMANENCY IN 12 MONTHS FOR CHILDREN IN CARE ON FIRST DAY 24 MONTHS
OR MORE
*****************************
*****************************

* This syntax is provided for informational purposes only. It requires
access to
* child-level data from all states, the District of Columbia, and Puerto
Rico.
* The following syntax is used to calculate an individual state's
performance
* for a recent cohort of children against the national standard and the
* historical cohorts (from all other states) that were used to establish
the
* national standard. Although this syntax will calculate RSPs and data
for all
* states, the only results of interest are for the state being evaluated
* (MyState "XX").

*****************************
* SPECIFY THE STATE AND 12-MONTH COHORT WHOSE PERFORMANCE IS BEING
ASSESSED
*****************************

* Note: For the 12-month cohort specified (CurrentCohort), there must
exist a
* file for that same cohort in "C:\cfsr3\Performance observed child".
This file
* will contain observed performance for a recent cohort of children (from
all
* states). For example, if the user specifies CurrentCohort = AB14, their
must
* be a file called "CFSR 3 - Observed perf for perm (FD 24 or more)
AB14.dta."

set more off

*****************************
* USER INPUT REQUIRED
*****************************

* Specify root folder where your files and folders are.
* This folder must have these subfolders:
* ... \Performance modeled
* ... \Performance observed child
* ... \Fixed files

local dirstub "C:\CFSR 3\Analysis - DP 2018 March"
```

```
* Specify 12-month cohort
local CurrentCohortList AB17
*local CurrentCohortList BA13 AB13 BA14 AB14 BA15 AB15 BA16 AB16 BA17


* Specify state(s) to run.
* Option 1 is to list them below in the local line (e.g.):
local statelist MI MS
local statetxtlist Michigan Mississippi
*      local statelist MA NH VT ME CT RI PA NY
* Option 2 is to list them in a dataset (which can be conveniently used
for the
* other indicators).
*use "`dirstub'/1 - Fixed files/States 2017.dta", clear
*local statelist
*local statetxtlist
*forvalues i = 1/`=_N' {
     *local statelist `"`statelist' `"`=stateabb[`i']'"'"'
      *local statetxtlist `"`statetxtlist' `"`=statetxt[`i']'"'"'
*}
****************************
* RUN SYNTAX FROM HERE TO END
****************************
foreach CurrentCohort of local CurrentCohortList {

local numelements : word count `statelist'
forvalue i = 1/`numelements'{
local MyState : word `i' of `statelist'
local MyStateTxt : word `i' of `statetxtlist'

* create log file.
log using "`dirstub'/4 - Performance modeled/RSP for perm (FD 24 or more)
`MyState' `CurrentCohort'.txt", text replace

****************************
* GET SOURCE FILE and PREP FILE
****************************

* Open the current cohort file. This file contains observed performance
for a
* recent cohort of children (from all states).
use "`dirstub'/3 - Performance observed child/CFSR 3 - Observed perf for
perm (FD 24 or more) `CurrentCohort'.dta", clear
keep if stateabb == `"`MyState'"'


* Flag if state requested is missing from the current cohort data file
(usually due to failing DQ).
quietly count
local DQState = 0
if r(N) == 0 {
      local DQState = 1
```

```
}

tempfile holding
save `holding'

* Open the historical cohort file. This file contains the fixed national
* standard for the indicator (Perf_Nation) and the observed performance
for the
* 12-month historical cohort of children that was used to establish the
national
* standard. We will be assessing the state's performance with its most
recent
* cohort (currently stored in 'holding') against the national standard
and
* historical cohorts from all other states. This syntax assumes the
historical
* cohort file is saved in the Fixes files folder of the dirstub path
defined earlier.

use "`dirstub'/1 - Fixed files/CFSR 3 - NS Observed perf for perm (FD 24
or more) BA14.dta", clear


quietly summarize ChildAge, detail
local ChildAge_Med = r(p50)
local Perf_NationNS = Perf_Nation[1]


* IF State requested does not have a current cohort score, creat a
special DQ
* output for that state rather than run the rest of the code.
if `DQState' == 1 {
      clear
      set obs 10
      generate stateabb = `"`MyState'"'
      generate state = `"`MyStateTxt'"'
      generate Indicator_new = "Permanency in 12 months (24+ mos)"
      generate NS = "" + string(`Perf_NationNS'*100,"%02.1f")+"%"
      generate TwelveMoCohort = `"`CurrentCohort'"'
            if substr(`"`CurrentCohort'"',1,2) ==  "AB"{
            gen Period = substr(`"`CurrentCohort'"',3,2) + "A" +
substr(`"`CurrentCohort'"',3,2) + "B"
            gen DataUsed = substr(`"`CurrentCohort'"',3,2) + "A-" +
substr(`"`CurrentCohort'"',3,2) + "B"
      }
      else if substr(`"`CurrentCohort'"',1,2) == "BA" {
            gen Period = string(real(substr(`"`CurrentCohort'"',3,2))-1)
+ "B" + substr(`"`CurrentCohort'"',3,2) + "A"
            gen DataUsed = string(real(substr(`"`CurrentCohort'"',3,2))-
1) + "B-" + substr(`"`CurrentCohort'"',3,2) + "A"
      }
      generate Format = "Percent"
      generate RSP_NS = ""
      generate RSP_Range = ""
```

```
      generate RSP_Range_Formatted = ""
      generate RSP_Pip = ""
      generate ValueType = "DataUsed2" in 1
      replace ValueType = "Den_State" in 2
      replace ValueType = "Num_State" in 3
      replace ValueType = "Perf_NationRnd" in 4
      replace ValueType = "Perf_StateRnd" in 5
      replace ValueType = "RSP_NS2" in 6
      replace ValueType = "RspLowRnd" in 7
      replace ValueType = "RSP_Range2" in 8
      replace ValueType = "RSPRnd" in 9
      replace ValueType = "RSPUppRnd" in 10
      generate Result = 0
      replace Result = round(`Perf_NationNS',0.001) in 4
      generate Result_String = "DQ"
      replace Result_String = DataUsed in 1
      replace Result_String = "" in 4
      replace Result_String = "" in 6
      generate MatchField = ""
      *Save file
      save "`dirstub'/4 - Performance modeled/CFSR 3 - RSP for perm (FD
24 or more) - `CurrentCohort' `MyState'.dta", replace
      outsheet stateabb state     Indicator_new NS TwelveMoCohort Period
DataUsed Format RSP_NS RSP_Range RSP_Range_Formatted RSP_Pip ValueType
Result Result_String MatchField ///
      using "`dirstub'/6 - Data for Tableau/Stata
Output/RSP_Perm12FD24_more_Summary `CurrentCohort' `MyState'.csv", comma
nolabel replace
      export excel stateabb state  Indicator_new NS TwelveMoCohort Period
DataUsed Format RSP_NS RSP_Range RSP_Range_Formatted RSP_Pip ValueType
Result Result_String MatchField ///
      using "`dirstub'/6 - Data for Tableau/Stata
Output/RSP_Perm12FD24_more_Summary `CurrentCohort' `MyState'.xlsx",
nolabel replace firstrow (variables)
      *close log and return to the top of the loop
       log close
       continue
}

drop if stateabb == `"`MyState'"'
append using `holding'

* For the state being evaluated, replace the value it has for national
observed
* performance (which is based on the current cohort for all states) with
the
* fixed NS for this indicator. The level of precision matters for the
* calculations RSP, RSPUpp, and RSPLow matters.


replace Perf_Nation = `Perf_NationNS' if stateabb == `"`MyState'"'

sort state
```

```
* Describe dataset
* Pick one observation per state
* Count and verify number of states in file
* Calculate desired direction for performance on this indicator
* Verify desired 12-month cohort has been selected

describe
egen pickone = tag(stateabb)
count if pickone
gen Passing = "Above NS"
tab Perf_Nation
tab TwelveMoCohort

****************************
* PREDICT OUTCOME
****************************

* Run multi-level logit model predicting permanency by 12 months
(Num_Child = 1)
* Adjust for child age on first day (ChildAge)

********  USER INPUT POSSIBLY NEEDED ********

* The xtmelogit command requires the user to enter the value of X in
* ib(X).ChildAge, where X represents the reference group for children in
* the national dataset. The median age is used as the reference group.
The ib
* function creates a dummy variable for each age value.
* To get the median age value run:

summarize ChildAge, detail

* The coded value to use is the one next to "50%." Enter that value in
ib(X)
* below.

timer on 1
xtmelogit Num_Child ib(`ChildAge_Med').ChildAge, baselevels || state:,
variance
timer off 1
timer list 1
timer clear

predict xb, xb
predict re, reffects
predict rese, reses

* xb = child's predicted log odds of permanency based on child's age, but
*       without considering the child's home state (i.e., ignoring the
state's
*       random effect)
* re = state's random effect (shift in child's predicted log odds of
*    permanency after considering the child's home state; aka, Empirical
*    Bayes intercept)
```

```
* rese - standard error of state's random effect

*****************************
* CALCULATE RISK-STANDARDIZED PERFORMANCE
*****************************

* 1. Calculate PREDICTED number of permanent exits in each state
*****************************

* The predicted number of permanent exits based on the state's
performance with
* its observed case mix. This is our best prediction of future
performance,
* assuming no change in case mix or policy. It is calculated as the sum
of each
* child's predicated probability of permanent exit, which is a complex
function
* of the child's value of xb and his or her state's specific random
effect.

sort state
gen double Child_Pred = exp(xb+re)/(1+exp(xb+re))
by state: egen double Num_Pred = total(Child_Pred)

* 2. Calculate EXPECTED number of permanenct exits in each state
*****************************

* The expected number of permanent exits based on the nation's
performance with
* the state's case mix. This represents how many permanent exits we would
expect
* for the state's children if they were treated in an "average" state. It
is
* calculated as the sum of each child's predicted probability of
permanent exit
* (xb), including the *average* intercept of all states (the average of
the
* random effects across states is zero).

gen double Child_Exp = exp(xb)/(1+exp(xb))
by state: egen double Num_Exp = total(Child_Exp)

* 3. Calculate risk-standardized ratio and risk-standardized performance
*****************************

* Calculate ratio of predicted to expected
* Multiply ratio by national observed performance (i.e., the national
standard)
* to get RSP
gen double Ratio_PE = Num_Pred / Num_Exp
gen double RSP = (Num_Pred / Num_Exp) * Perf_Nation

* 4. Calculate 95% confidence intervals for RSP
*****************************
```

```
* Upper CI
sort state
gen double UppNum = exp(xb+re+(1.96*rese))/(1+exp(xb+re+(1.96*rese)))
by state: egen double UppNumSum = total(UppNum)
gen double UppDen = exp(xb)/(1+exp(xb))
by state: egen double UppDenSum = total(UppDen)
gen double RSPUpp = (UppNumSum/UppDenSum) * Perf_Nation

* Lower CI
gen double LowNum = exp(xb+re-(1.96*rese))/(1+exp(xb+re-(1.96*rese)))
by state: egen double LowNumSum = total(LowNum)
gen double LowDen = exp(xb)/(1+exp(xb))
by state: egen double LowDenSum = total(LowDen)
gen double RSPLow = (LowNumSum/LowDenSum) * Perf_Nation

***************************
* COMPARE STATE'S RSP TO NATIONAL STANDARD (NS)
***************************

* When comparing the state's RSP (actually, the CIs) to the national
observed
* performance, use rounded versions.

* Round the national observed performance
clonevar Perf_NationRnd = Perf_Nation
replace Perf_NationRnd = round(Perf_NationRnd,0.001)

* Round the CIs of the RSP
clonevar RSPLowRnd = RSPLow
clonevar RSPUppRnd = RSPUpp
replace RSPLowRnd = round(RSPLowRnd,0.001)
replace RSPUppRnd = round(RSPUppRnd,0.001)

* Compare CI's of the RSP relative to national observed performance
gen RSP_NS = "No diff"
replace RSP_NS = "Met" if RSPLowRnd > Perf_NationRnd
replace RSP_NS = "Not met" if RSPUppRnd < Perf_NationRnd

* Count number of states meeting, not meeting, and no different from NS
egen RSP_Met=total(RSP_NS=="Met" & pickone)
egen RSP_NotMet=total(RSP_NS=="Not met" & pickone)
egen RSP_NotDif=total(RSP_NS=="No diff" & pickone)

* Count number of states that must engage in a PIP
gen RSP_Pip = ""
replace RSP_Pip = "No PIP" if RSP_NS == "Met"
replace RSP_Pip = "No PIP" if RSP_NS == "No diff"
replace RSP_Pip = "PIP" if RSP_NS == "Not met"

* Save
save "`dirstub'/4 - Performance modeled/CFSR 3 - RSP for perm (FD 24 or
more) - `CurrentCohort' `MyState'.dta", replace
```

```
**************************
* REPORTING
**************************

* Export to excel the state's results, formatted for profile

gen Indicator_new = "Permanency in 12 months (24+ mos)"

* Round the RSP and Perf_State to 1 decimal (so it's, e.g., .421; avoids
having
* to format it later in Excel to show only 1 decimal)
clonevar RSPRnd = RSP
replace RSPRnd = round(RSPRnd,0.001)
clonevar Perf_StateRnd = Perf_State
replace Perf_StateRnd = round(Perf_StateRnd,0.001)

gen RSP_Range = string(RSPLowRnd*100,"%02.1f") + "-" +
string(RSPUppRnd*100,"%02.1f")
gen RSP_Range_Formatted = string(RSPLowRnd*100,"%02.1f") + "%-" +
string(RSPUppRnd*100,"%02.1f") + "%"
gen Format = "Percent"
if substr(`"`CurrentCohort'"',1,2) ==  "AB"{
      gen Period = substr(`"`CurrentCohort'"',3,2) + "A" +
substr(`"`CurrentCohort'"',3,2) + "B"
      gen DataUsed = substr(`"`CurrentCohort'"',3,2) + "A-" +
substr(`"`CurrentCohort'"',3,2) + "B"
}
else if substr(`"`CurrentCohort'"',1,2) == "BA" {
      gen Period = string(real(substr(`"`CurrentCohort'"',3,2))-1) + "B"
+ substr(`"`CurrentCohort'"',3,2) + "A"
      gen DataUsed = string(real(substr(`"`CurrentCohort'"',3,2))-1) +
"B-" + substr(`"`CurrentCohort'"',3,2) + "A"
}
* MH These variables only exist to ensure that there is both a column and
record with this data after reshaping below, and because tableau.
gen NS = string(`Perf_NationNS'*100,"%02.1f")+"%"
gen RSP_Range2 = RSP_Range_Formatted
gen RSP_NS2 = RSP_NS
gen DataUsed2 = DataUsed
gen MatchField = ""

* gen RSP_new = round(RSP,0.001)*100 (e.g., "40.1")
* gen RSPLowRnd_new = RSPLowRnd*100
* gen RSPUppRnd_new = RSPUppRnd*100
* gen RSPRange = string(RSPLowRnd_new,"%02.1f") + " - " +
string(RSPUppRnd_new,"%02.1f")
* gen Perf_NationRnd_new = Perf_NationRnd*100

* outsheet stateabb Indicator_new TwelveMoCohort RSP_new RSPRange
Perf_NationRnd_new RSP_NS RSP_Pip ///
* if (stateabb == `"`MyState'"' & pickone==1) using
"Profile_RSP_Perm12FD24_more_Summary `CurrentCohort' `MyState'.csv",
comma nolabel replace
```

```
outsheet stateabb statetxt Indicator_new NS TwelveMoCohort Period
DataUsed DataUsed2 Format RSP_NS RSP_NS2 RSP_Range RSP_Range_Formatted
RSP_Range2 RSP_Pip Den_State Num_State Perf_StateRnd RSPLowRnd RSPRnd
RSPUppRnd Perf_NationRnd MatchField ///
if (stateabb == `"`MyState'"' & pickone==1) using "`dirstub'/4 -
Performance modeled/RSP_Perm12FD24_more_Summary `CurrentCohort'
`MyState'.csv", comma nolabel replace


import delimited "`dirstub'/4 - Performance
modeled/RSP_Perm12FD24_more_Summary `CurrentCohort' `MyState'.csv", clear
case(preserve)

rename (Den_State Num_State Perf_StateRnd RSPLowRnd RSPRnd RSPUppRnd
Perf_NationRnd) Result=
rename (RSP_Range2 RSP_NS2 DataUsed2) str=

reshape long Result str, i(stateabb statetxt) j(ValueType) string
*reshape long result str, i(stateabb statetxt indicator_new ns
twelvemocohort period data_used format rsp_ns rsp_range
rsp_range_formatted rsp_pip) j(ValueType) string
replace Result = 0 if missing(Result)


rename (str) Result_String
rename (statetxt) state


outsheet stateabbstate Indicator_new    NS    TwelveMoCohort   Period
      DataUsed    Format      RSP_NS      RSP_Range   RSP_Range_Formatted
      RSP_Pip     ValueType   Result      Result_String MatchField ///
 using "`dirstub'/6 - Data for Tableau/Stata
Output/RSP_Perm12FD24_more_Summary `CurrentCohort' `MyState'.csv", comma
nolabel replace


export excel stateabb state  Indicator_new NS TwelveMoCohort Period
DataUsed Format RSP_NS RSP_Range RSP_Range_Formatted RSP_Pip ValueType
Result Result_String MatchField ///
 using "`dirstub'/6 - Data for Tableau/Stata
Output/RSP_Perm12FD24_more_Summary `CurrentCohort' `MyState'.xlsx",
nolabel replace firstrow (variables)


log close

}
}
```

CFSR 3: RISK-STANDARDIZED PERFORMANCE - PLACEMENT STABILITY

```
* VERSION HISTORY
*****************************

*****************************
*****************************
* CFSR 3: RISK-STANDARDIZED PERFORMANCE
* - PLACEMENT STABILITY
*****************************
*****************************

* This syntax is provided for informational purposes only. It requires
access to
* child-level data from all states, the District of Columbia, and Puerto
Rico.
* The following syntax is used to calculate an individual state's
performance
* for a recent cohort of children against the national standard and the
* historical cohorts (from all other states) that were used to establish
the
* national standard. Although this syntax will calculate RSPs and data
for all
* states, the only results of interest are for the state being evaluated
* (MyState "XX").

*****************************
* SPECIFY THE STATE AND 12-MONTH COHORT WHOSE PERFORMANCE IS BEING
ASSESSED
*****************************

* Note: For the 12-month cohort specified (CurrentCohort), there must
exist a
* file for that same cohort in "C:\cfsr3\Performance observed child".
This file
* will contain observed performance for a recent cohort of children (from
all
* states). For example, if the user specifies CurrentCohort = AB14, their
must
* be a file called "CFSR 3 - Observed perf for placement stability
AB14.dta."

set more off
set trace off
*set min_memory 16g

*****************************
* USER INPUT REQUIRED
*****************************

* Specify root folder where your files and folders are.
* This folder must have these subfolders:
* ... \Performance modeled
* ... \Performance observed child
* ... \Fixed files
```

```
local dirstub "C:/CFSR 3/Analysis - DP 2018 March"

* Specify 12-month cohort
* local CurrentCohortList BA10 AB10 BA11 AB11 BA12 AB12 BA13 AB13 BA14
AB14 BA15 AB15 BA16 AB16
 local CurrentCohortList AB17
* Specify state(s) to run.
* Option 1 is to list them below in the local line:
local statelist MI MS
local statetxtlist Michigan Mississippi
* Option 2 is to list them in a dataset (which can be conveniently used
for the
* other indicators).
*use "`dirstub'/1 - Fixed files/States 2017.dta", clear
* levelsof stateabb, local(statelist)
* levelsof statetxt, local(statetxtlist)
*local statelist
*local statetxtlist
*forvalues i = 1/`=_N' {
     *local statelist `"`statelist' `"`=stateabb[`i']'"'"'
      *local statetxtlist `"`statetxtlist' `"`=statetxt[`i']'"'"'
*}
****************************
* RUN SYNTAX FROM HERE TO END
****************************
foreach CurrentCohort of local CurrentCohortList {

local numelements : word count `statelist'
*forvalue i = 12/14{
forvalue i = 1/`numelements'{
local MyState : word `i' of `statelist'
local MyStateTxt : word `i' of `statetxtlist'

* create log file.
log using "`dirstub'/4 - Performance modeled/RSP for placement stability
`MyState' `CurrentCohort'.txt", text replace

****************************
* GET SOURCE FILE and PREP FILE
****************************

* Open the current cohort file. This file contains observed performance
for a
* recent cohort of children (from all states).
use "`dirstub'/3 - Performance observed child/CFSR 3 - Observed perf for
placement stability `CurrentCohort'.dta", clear
keep if stateabb == `"`MyState'"'

* Flag if state requested is missing from the current cohort data file
(usually due to failing DQ).
quietly count
local DQState = 0
if r(N) == 0 {
      local DQState = 1
```

```
}

tempfile holding
save `holding'

* Open the historical cohort file. This file contains the fixed national
* standard for the indicator (Perf_Nation) and the observed performance
for the
* 12-month historical cohort of children that was used to establish the
national
* standard. We will be assessing the state's performance with its most
recent
* cohort (currently stored in 'holding') against the national standard
and
* historical cohorts from all other states. This syntax assumes the
historical
* cohort file is saved in the Fixes files folder of the dirstub path
defined earlier.

use "`dirstub'/1 - Fixed files/CFSR 3 - NS Observed perf for placement
stability BA14.dta", clear

quietly summarize ChildAge, detail
local ChildAge_Med = r(p50)
local Perf_NationNS = Perf_Nation[1]
local Perf_NationNS_MP = Perf_Nation_MP[1]

* IF State requested does not have a current cohort score, creat a
special DQ
* output for that state rather than run the rest of the code.
if `DQState' == 1 {
      clear
      set obs 10
      generate stateabb = `"`MyState'"'
      generate state = `"`MyStateTxt'"'
      generate Indicator_new = "Placement stability (moves/1,000 days in
care)"
      generate NS = string(round(`Perf_NationNS_MP',0.01))
      generate TwelveMoCohort = "`CurrentCohort'"
if substr(`"`CurrentCohort'"',1,2) ==  "AB"{
               gen Period = substr(`"`CurrentCohort'"',3,2) + "A" +
substr(`"`CurrentCohort'"',3,2) + "B"
}
else if substr(`"`CurrentCohort'"',1,2) == "BA" {
               gen Period = string(real(substr(`"`CurrentCohort'"',3,2))-
1) + "B" + substr(`"`CurrentCohort'"',3,2) + "A"
}

      generate DataUsed = Period
      generate Format = "Number"
      generate RSP_NS = ""
      generate RSP_Range = ""
      generate RSP_Range_Formatted = ""
      generate RSP_Pip = ""
```

```
      generate ValueType = "DataUsed2" in 1
      replace ValueType = "Den_State" in 2
      replace ValueType = "Num_State" in 3
      replace ValueType = "Perf_NationRnd" in 4
      replace ValueType = "Perf_StateRnd" in 5
      replace ValueType = "RSP_NS2" in 6
      replace ValueType = "RspLowRnd" in 7
      replace ValueType = "RSP_Range2" in 8
      replace ValueType = "RSPRnd" in 9
      replace ValueType = "RSPUppRnd" in 10
      generate Result = 0
      replace Result = round(`Perf_NationNS_MP',0.01) in 4
      generate Result_String = "DQ"
      replace Result_String = Period in 1
      replace Result_String = "" in 4
      replace Result_String = "" in 6
      generate MatchField = ""
      *Save file
      save "`dirstub'/4 - Performance modeled/CFSR 3 - RSP
for_PlacementStability_Summary `CurrentCohort' `MyState'.dta", replace
      outsheet stateabb state     Indicator_new NS TwelveMoCohort Period
DataUsed Format RSP_NS RSP_Range RSP_Range_Formatted RSP_Pip ValueType
Result Result_String MatchField ///
      using "`dirstub'/6 - Data for Tableau/Stata
Output/RSP_PlacementStability_Summary `CurrentCohort' `MyState'.csv",
comma nolabel replace
      export excel stateabb state  Indicator_new NS TwelveMoCohort Period
DataUsed Format RSP_NS RSP_Range RSP_Range_Formatted RSP_Pip ValueType
Result Result_String MatchField ///
      using "`dirstub'/6 - Data for Tableau/Stata
Output/RSP_PlacementStability_Summary `CurrentCohort' `MyState'.xlsx",
nolabel replace firstrow (variables)
      *close log and return to the top of the loop
       log close
       continue
}


drop if stateabb == `"`MyState'"'
append using `holding'

* For the state being evaluated, replace the value it has for national
observed
* performance (which is based on the current cohort for all states) with
the
* fixed NS for this indicator. The level of precision matters for the
* calculations RSP, RSPUpp, and RSPLow matters.

* IMPORTANT - For Rate indicators (PS & MALTX FC), replace Perf_Nation_MP
as
* well. For the Rate indicators, the _MP version is used as the NS when
* comparing to the RSP.

replace Perf_Nation_MP = `Perf_NationNS_MP' if stateabb == `"`MyState'"'
```

```
replace Perf_Nation = `Perf_NationNS' if stateabb == `"`MyState'"'

sort state

* Describe dataset
* Pick one observation per state
* Count and verify number of states in file
* Calculate desired direction for performance on this indicator
* Verify desired 12-month cohort has been selected

describe
egen pickone = tag(stateabb)
count if pickone
gen Passing = "Below NS"
tab Perf_Nation_MP
tab Perf_Nation
tab TwelveMoCohort

****************************
* PREDICT OUTCOME
****************************

* Run multi-level poisson model predicting placement moves (Num_Child)
given
* one or more foster care episodes with a particular duration in days
* (Den_Child). Adjust for child age at entry (ChildAge).

********  POSSIBLE USER INPUT NEEDED ********

* The xtmepoisson command requires the user to enter the value of X in
* ib(X).ChildAge, where X represents the reference group for children in
the
* national dataset. The median age is used as the reference group. The ib
* function creates a dummy variable for each age value.
* To get the median age value run:

summarize ChildAge, detail

* The coded value to use is the one next to "50%." Enter that value in
ib(X)
* below.

timer on 1
capture noisily xtmepoisson Num_Child ib(`ChildAge_Med').ChildAge,
baselevels exposure(Den_Child) || state:, variance iterate(20)
timer off 1
timer list 1
timer clear

predict xb, xb
predict re, reffects
predict rese, reses
```

```
* xb = natural logarithm of child's predicted number of placement moves
based on
* child's age and the number of days of the child's foster care
episode(s)
* (i.e., exposure to the event of interest), but without considering the
child's
* home state (i.e., ignoring the state's random effect)

* re = state's random effect (shift in the natural logarithm of the
child's
* predicted number of placement moves after considering the child's home
state;
* aka, Empirical Bayes intercept)

* rese - standard error of state's random effect

****************************
* CALCULATE RISK-STANDARDIZED PERFORMANCE
****************************

* 1. Calculate PREDICTED number of placement moves in each state
****************************

* The predicted number of placement moves based on the state's
performance with
* its observed case mix. This is our best prediction of future
performance,
* assuming no change in case mix or policy. It is calculated as the sum
of each
* child's predicted number of placement moves, which is a complex
function of
* the child's value of xb (based on child's age and episode duration in
days)
* and his or her state's specific random effect).

sort state
gen double Child_Pred = exp(xb+re)
by state: egen double Num_Pred = total(Child_Pred)

* 2. Calculate EXPECTED number of placement moves in each state
****************************

* The expected number of placement moves based on the nation's
performance with
* the state's case mix. This represents how many placement moves we would
expect
* for the state's children if they were treated in an "average" state. It
is
* calculated as the sum of each child's predicted number of placement
moves (xb),
* including the *average* intercept of all states (the average of the
* random effects across states is zero).

gen double Child_Exp = exp(xb)
```

```
by state: egen double Num_Exp = total(Child_Exp)

* 3. Calculate risk-standardized ratio and risk-standardized performance
*****************************

* Calculate ratio of predicted to expected
* Multiply ratio by national observed performance (i.e., the national
standard)
* to get RSP
gen double Ratio_PE = Num_Pred / Num_Exp
gen double RSP = (Num_Pred / Num_Exp) * Perf_Nation
gen double RSP_MP = RSP * 1000

* 4. Calculate 95% confidence intervals for RSP
*****************************

* Upper CI of the RSP
sort state
gen double UppNum = exp(xb+re+(1.96*rese))
by state: egen double UppNumSum = total(UppNum)
gen double UppDen = exp(xb)
by state: egen double UppDenSum = total(UppDen)
gen double RSPUpp = (UppNumSum/UppDenSum) * Perf_Nation
gen double RSPUpp_MP = RSPUpp * 1000

* Low CI of the RSP
gen double LowNum = exp(xb+re-(1.96*rese))
by state: egen double LowNumSum = total(LowNum)
gen double LowDen = exp(xb)
by state: egen double LowDenSum = total(LowDen)
gen double RSPLow = (LowNumSum/LowDenSum) * Perf_Nation
gen double RSPLow_MP = RSPLow * 1000

*****************************
* COMPARE STATE'S RSP TO NATIONAL STANDARD
*****************************

* When comparing the state's RSP (actually, the CIs) to the national
observed
* performance, use rounded versions. For rate indicators (maltreatment in
* foster care and placement stability), use Perf_Nation_MP, RSRLow_MP,
etc.
* instead of the unmultiplied rate (e.g., Perf_Nation, RSRLow, which have
too
* many trailing 0s such that the rounded rates will all = 0; e.g.,
.00000631
* will get rounded to .00; whereas 6.3094187 will get rounded to 6.31).

* Round the national observed performance
clonevar Perf_NationRnd = Perf_Nation_MP
replace Perf_NationRnd = round(Perf_NationRnd,0.01)

* Round the CIs of the RSP
clonevar RSPLowRnd = RSPLow_MP
```

```
clonevar RSPUppRnd = RSPUpp_MP
replace RSPLowRnd = round(RSPLowRnd,0.01)
replace RSPUppRnd = round(RSPUppRnd,0.01)

* Compare CI's of the RSP relative to national observed performance
gen RSP_NS = "No diff"
replace RSP_NS = "Met" if RSPUppRnd < Perf_NationRnd
replace RSP_NS = "Not met" if RSPLowRnd > Perf_NationRnd

* Count number of states meeting, not meeting, and no different from NS
egen RSP_Met=total(RSP_NS=="Met" & pickone)
egen RSP_NotMet=total(RSP_NS=="Not met" & pickone)
egen RSP_NotDif=total(RSP_NS=="No diff" & pickone)

* Count number of states that must engage in a PIP
gen RSP_Pip = ""
replace RSP_Pip = "No PIP" if RSP_NS == "Met"
replace RSP_Pip = "No PIP" if RSP_NS == "No diff"
replace RSP_Pip = "PIP" if RSP_NS == "Not met"



* Create variable holding median age at entry to include in output. The
coded
* age values are one higher than the actual age (e.g., a coded value of 7
* represents an actual age of 6), so median age = coded value for median
age
* minus 1.
sort state
by state: egen MedAge = median(ChildAge)
replace MedAge = MedAge -1

* Save
save "`dirstub'/4 - Performance modeled/CFSR 3 - RSP for placement
stability - `CurrentCohort' `MyState'.dta", replace

****************************
* REPORTING
****************************

* Export to excel the state's results, formatted for profile

gen Indicator_new = "Placement stability (moves/1,000 days in care)"

* Round the RSP and Perf_State to 2 decimals (so it's, e.g., 3.57; avoids
having
* to format it later in Excel to show only 2 decimals)
gen RSPRnd = round(RSP_MP,0.01)
gen Perf_StateRnd = round(Perf_State_MP,0.01)

* gen RSP_new = round(RSP,0.001)*100 (e.g., "40.1")
gen RSP_Range = string(RSPLowRnd) + "-" + string(RSPUppRnd)
gen RSP_Range_Formatted = string(RSPLowRnd) + "-" + string(RSPUppRnd)
gen Format = "Number"
```

```
if substr(`"`CurrentCohort'"',1,2) ==  "AB"{
               gen Period = substr(`"`CurrentCohort'"',3,2) + "A" +
substr(`"`CurrentCohort'"',3,2) + "B"
}
else if substr(`"`CurrentCohort'"',1,2) == "BA" {
               gen Period = string(real(substr(`"`CurrentCohort'"',3,2))-
1) + "B" + substr(`"`CurrentCohort'"',3,2) + "A"
}

if substr(`"`CurrentCohort'"',1,2) ==  "AB"{
               gen DataUsed = substr(`"`CurrentCohort'"',3,2) + "A-" +
substr(`"`CurrentCohort'"',3,2) + "B"
}
else if substr(`"`CurrentCohort'"',1,2) == "BA" {
               gen DataUsed =
string(real(substr(`"`CurrentCohort'"',3,2))-1) + "B-" +
substr(`"`CurrentCohort'"',3,2) + "A"
}

* MH These variables only exist to ensure that there is both a column and
record with this data after reshaping below, and because tableau.
gen NS = string(Perf_NationRnd)
gen RSP_Range2 = RSP_Range_Formatted
gen RSP_NS2 = RSP_NS
gen DataUsed2 = DataUsed
gen MatchField = ""

* RSP (e.g., .0035744702...)
* RSP_MP (e.g., 3.5744702)
* gen RSP_new = round(RSP_MP,0.01) (e.g., "3.57")
* gen RSPRange = string(RSPLowRnd) + " - " + string(RSPUppRnd)
*(e.g., "3.49 - 3.67")
* Perf_State_MP (e.g., 3.5348582)

* outsheet stateabb Indicator_new TwelveMoCohort RSP_new RSPRange
Perf_NationRnd RSP_NS RSP_Pip ///
* if (stateabb == `"`MyState'"' & pickone==1) using "`dirstub'/4 -
Performance modeled/Profile_RSP_PlacementStability `CurrentCohort'
`MyState'.csv", comma nolabel replace

outsheet stateabb statetxt Indicator_new NS TwelveMoCohort Period
DataUsed DataUsed2 Format RSP_NS RSP_NS2 RSP_Range RSP_Range_Formatted
RSP_Range2 RSP_Pip Den_State Num_State Perf_StateRnd RSPLowRnd RSPRnd
RSPUppRnd Perf_NationRnd MatchField ///
if (stateabb == `"`MyState'"' & pickone==1) using "`dirstub'/4 -
Performance modeled/RSP_PlacementStability_Summary `CurrentCohort'
`MyState'.csv", comma nolabel replace


import delimited "`dirstub'/4 - Performance
modeled/RSP_PlacementStability_Summary `CurrentCohort' `MyState'.csv",
clear case(preserve)
```

```
rename (Den_State Num_State Perf_StateRnd RSPLowRnd RSPRnd RSPUppRnd
Perf_NationRnd) Result=
rename (RSP_Range2 RSP_NS2 DataUsed2) str=

reshape long Result str, i(stateabb statetxt) j(ValueType) string
*reshape long result str, i(stateabb statetxt indicator_new ns
twelvemocohort period data_used format rsp_ns rsp_range
rsp_range_formatted rsp_pip) j(ValueType) string
replace Result = 0 if missing(Result)


rename (str) Result_String
rename (statetxt) state

outsheet stateabbstate Indicator_new    NS    TwelveMoCohort   Period
      DataUsed    Format      RSP_NS      RSP_Range   RSP_Range_Formatted
      RSP_Pip     ValueType   Result      Result_String MatchField ///
 using "`dirstub'/6 - Data for Tableau/Stata
Output/RSP_PlacementStability_Summary `CurrentCohort' `MyState'.csv",
comma nolabel replace

export excel stateabb state  Indicator_new NS TwelveMoCohort Period
DataUsed Format RSP_NS RSP_Range RSP_Range_Formatted RSP_Pip ValueType
Result Result_String MatchField ///
 using "`dirstub'/6 - Data for Tableau/Stata
Output/RSP_PlacementStability_Summary `CurrentCohort' `MyState'.xlsx",
nolabel replace firstrow (variables)

*close log and return to the top of the loop
 log close
}
}
```

CFSR 3: RISK-STANDARDIZED PERFORMANCE - RE-ENTRY TO FOSTER CARE IN 12 MONTHS

```
* VERSION HISTORY
*****************************

*****************************
*****************************
* CFSR 3: RISK-STANDARDIZED PERFORMANCE
* - RE-ENTRY TO FOSTER CARE IN 12 MONTHS
*****************************
*****************************

* This syntax is provided for informational purposes only. It requires
access to
* child-level data from all states, the District of Columbia, and Puerto
Rico.
* The following syntax is used to calculate an individual state's
performance
* for a recent cohort of children against the national standard and the
* historical cohorts (from all other states) that were used to establish
the
* national standard. Although this syntax will calculate RSPs and data
for all
* states, the only results of interest are for the state being evaluated
* (MyState "XX").

*****************************
* SPECIFY THE STATE AND 12-MONTH COHORT WHOSE PERFORMANCE IS BEING
ASSESSED
*****************************

* Note: For the 12-month cohort specified (CurrentCohort), there must
exist a
* file for that same cohort in "C:\cfsr3\Performance observed child".
This file
* will contain observed performance for a recent cohort of children (from
all
* states). For example, if the user specifies CurrentCohort = AB12, their
must
* be a file called "CFSR 3 - Observed perf for reentry AB12.dta."

set more off
set trace off
*set min_memory 16g

*****************************
* USER INPUT REQUIRED
*****************************

* Specify root folder where your files and folders are.
* This folder must have these subfolders:
* ... \Performance modeled
* ... \Performance observed child
* ... \Fixed files

local dirstub "C:/CFSR 3/Analysis - DP 2018 March"
```

```
* Specify 12-month cohort
local CurrentCohortList  AB15 BA16
*
* Specify state(s) to run.
* Option 1 is to list them below in the local line:
 local statelist MI MS
 local statetxtlist Michigan Mississippi

* Option 2 is to list them in a dataset (which can be conveniently used
for the
* other indicators).
*use "`dirstub'/1 - Fixed files/States 2017.dta", clear
*local statelist
*local statetxtlist
*forvalues i = 1/`=_N' {
     *local statelist `"`statelist' `"`=stateabb[`i']'"'"'
      *local statetxtlist `"`statetxtlist' `"`=statetxt[`i']'"'"'
 *}

****************************
* RUN SYNTAX FROM HERE TO END
****************************
macro list
foreach CurrentCohort of local CurrentCohortList {

local numelements : word count `statelist'

*forvalue i = 1/1{
forvalue i = 1/`numelements'{
local MyState : word `i' of `statelist'
local MyStateTxt : word `i' of `statetxtlist'

* create log file.
log using "`dirstub'/4 - Performance modeled/RSP for reentry `MyState'
`CurrentCohort'.txt", text replace

****************************
* GET SOURCE FILE and PREP FILE
****************************

* Open the current cohort file. This file contains observed performance
for a
* recent cohort of children (from all states).
use "`dirstub'/3 - Performance observed child/CFSR 3 - Observed perf for
reentry `CurrentCohort'.dta", clear
keep if stateabb == `"`MyState'"'

* Flag if state requested is missing from the current cohort data file
(usually due to failing DQ).
quietly count
local DQState = 0
if r(N) == 0 {
      local DQState = 1
```

```
}

tempfile holding
save `holding'

* Open the historical cohort file. This file contains the fixed national
* standard for the indicator (Perf_Nation) and the observed performance
for the
* 12-month historical cohort of children that was used to establish the
national
* standard. We will be assessing the state's performance with its most
recent
* cohort (currently stored in 'holding') against the national standard
and
* historical cohorts from all other states. This syntax assumes the
historical
* cohort file is saved in the Fixes files folder of the dirstub path
defined earlier.

use "`dirstub'/1 - Fixed files/CFSR 3 - NS Observed perf for reentry
BA12.dta", clear


quietly summarize ChildAge, detail
local ChildAge_Med = r(p50)
local Perf_NationNS = Perf_Nation[1]


* IF State requested does not have a current cohort score, creat a
special DQ
* output for that state rather than run the rest of the code.
if `DQState' == 1 {
      clear
      set obs 10
      generate stateabb = `"`MyState'"'
      generate state = `"`MyStateTxt'"'
      generate Indicator_new = "Re-entry to foster care"
      generate NS = "" + string(`Perf_NationNS'*100,"%02.1f")+"%"
      generate TwelveMoCohort = `"`CurrentCohort'"'
      if substr(`"`CurrentCohort'"',1,2) ==  "AB"{
                  gen Period = substr(`"`CurrentCohort'"',3,2) + "A" +
substr(`"`CurrentCohort'"',3,2) + "B"
                        gen DataUsed = substr(`"`CurrentCohort'"',3,2) + "A-"
+ string(real(substr(`"`CurrentCohort'"',3,2))+2) + "B"
}
else if substr(`"`CurrentCohort'"',1,2) == "BA" {
                  gen Period = string(real(substr(`"`CurrentCohort'"',3,2))-
1) + "B" + substr(`"`CurrentCohort'"',3,2) + "A"
                        gen DataUsed =
string(real(substr(`"`CurrentCohort'"',3,2))-1) + "B-" +
string(real(substr(`"`CurrentCohort'"',3,2))+2) + "A"
}
      generate Format = "Percent"
      generate RSP_NS = ""
```

```
      generate RSP_Range = ""
      generate RSP_Range_Formatted = ""
      generate RSP_Pip = ""
      generate ValueType = "DataUsed2" in 1
      replace ValueType = "Den_State" in 2
      replace ValueType = "Num_State" in 3
      replace ValueType = "Perf_NationRnd" in 4
      replace ValueType = "Perf_StateRnd" in 5
      replace ValueType = "RSP_NS2" in 6
      replace ValueType = "RspLowRnd" in 7
      replace ValueType = "RSP_Range2" in 8
      replace ValueType = "RSPRnd" in 9
      replace ValueType = "RSPUppRnd" in 10
      generate Result = 0
      replace Result = round(`Perf_NationNS',0.001) in 4
      generate Result_String = "DQ"
      replace Result_String = DataUsed in 1
      replace Result_String = "" in 4
      replace Result_String = "" in 6
      generate MatchField = ""
      *Save file
      save "`dirstub'/4 - Performance modeled/CFSR 3 - RSP for reentry -
`CurrentCohort' `MyState'.dta", replace
      outsheet stateabb state     Indicator_new NS TwelveMoCohort Period
DataUsed Format RSP_NS RSP_Range RSP_Range_Formatted RSP_Pip ValueType
Result Result_String MatchField ///
      using "`dirstub'/6 - Data for Tableau/Stata
Output/RSP_Reentry_Summary `CurrentCohort' `MyState'.csv", comma nolabel
replace
      export excel stateabb state  Indicator_new NS TwelveMoCohort Period
DataUsed Format RSP_NS RSP_Range RSP_Range_Formatted RSP_Pip ValueType
Result Result_String MatchField ///
      using "`dirstub'/6 - Data for Tableau/Stata
Output/RSP_Reentry_Summary `CurrentCohort' `MyState'.xlsx", nolabel
replace firstrow (variables)
      *close log and return to the top of the loop
       log close
       continue
}

drop if stateabb == `"`MyState'"'
append using `holding'

* For the state being evaluated, replace the value it has for national
observed
* performance (which is based on the current cohort for all states -
.083279099181) with the
* fixed NS for this indicator. The level of precision matters for the
* calculations RSP, RSPUpp, and RSPLow matters.

replace Perf_Nation = `Perf_NationNS' if stateabb == `"`MyState'"'

sort state
```

```
* Describe dataset
* Pick one observation per state
* Count and verify number of states in file
* Calculate desired direction for performance on this indicator
* Verify desired 12-month cohort has been selected

describe
egen pickone = tag(stateabb)
count if pickone
gen Passing = "Below NS"
tab Perf_Nation
tab TwelveMoCohort

****************************
* PREDICT OUTCOME
****************************

* Run multi-level logit model predicting re-entry by 12 months
(NumRE_Child = 1)
* Adjust for child age at exit (ChildAge) and state's entry rate
(EntryRate)
* The median age is used as the reference group.

xtmelogit NumRE_Child ib(`ChildAge_Med').ChildAge EntryRate, baselevels
|| state:, variance

predict xb, xb
predict re, reffects
predict rese, reses

* xb = child's predicted log odds of re-entry based on child's age and
the
*    state-specific entry rate, but without considering the child's home
state
*    (i.e., ignoring the state's random effect)
* re = state's random effect (shift in child's predicted log odds of
*    re-entry after considering the child's home state; aka, Empirical
*    Bayes intercept)
* rese - standard error of state's random effect

****************************
* CALCULATE RISK-STANDARDIZED PERFORMANCE
****************************

* 1. Calculate PREDICTED number of outcomes
****************************

* The predicted number of re-entries based on state's performance with
its
* observed case mix. This is our best prediction of future performance,
* assuming no change in case mix or policy. It is calculated as the sum
of each
* child's predicated probability of re-entry, which is a complex function
of the child's value of
```

```
* xb and his or her state's specific random effect.

sort state
gen double Child_Pred = exp(xb+re)/(1+exp(xb+re))
by state: egen double Num_Pred = total(Child_Pred)

* 2. Calculate EXPECTED number of outcomes
*****************************

* The expected number of re-entries based on the nation's performance
with
* the state's case mix. This represents how many re-entries we would
expect
* for the state's children if they were treated in an "average" state. It
is
* calculated as the sum of each child's predicted probability of re-entry
* (xb), including the *average* intercept of all states (the average of
the
* random effects across states is zero).

gen double Child_Exp = exp(xb)/(1+exp(xb))
by state: egen double Num_Exp = total(Child_Exp)

* 3. Calculate risk-standardized ratio and risk-standardized performance
*****************************

* Calculate ratio of predicted to expected
* Multiply ratio by national observed performance (i.e., the national
standard)
* to get RSP
gen double Ratio_PE = Num_Pred / Num_Exp
gen double RSP = (Num_Pred / Num_Exp) * Perf_Nation

* 4. Calculate 95% confidence intervals for RSP
*****************************

* Upper CI
sort state
gen double UppNum = exp(xb+re+(1.96*rese))/(1+exp(xb+re+(1.96*rese)))
by state: egen double UppNumSum = total(UppNum)
gen double UppDen = exp(xb)/(1+exp(xb))
by state: egen double UppDenSum = total(UppDen)
gen RSPUpp = (UppNumSum/UppDenSum) * Perf_Nation

* Lower CI
gen double LowNum = exp(xb+re-(1.96*rese))/(1+exp(xb+re-(1.96*rese)))
by state: egen double LowNumSum = total(LowNum)
gen double LowDen = exp(xb)/(1+exp(xb))
by state: egen double LowDenSum = total(LowDen)
gen double RSPLow = (LowNumSum/LowDenSum) * Perf_Nation

*****************************
* COMPARE STATE'S RSP TO NATIONAL STANDARD
*****************************
```

```
* When comparing the state's RSP (actually, the CIs) to the national
observed
* performance, use rounded versions.

* Round the national observed performance
clonevar Perf_NationRnd = Perf_Nation
replace Perf_NationRnd = round(Perf_NationRnd,0.001)

* Round the CIs of the RSP
clonevar RSPLowRnd = RSPLow
clonevar RSPUppRnd = RSPUpp
replace RSPLowRnd = round(RSPLowRnd,0.001)
replace RSPUppRnd = round(RSPUppRnd,0.001)

* Compare CI's of the RSP relative to national observed performance
gen RSP_NS = "No diff"
replace RSP_NS = "Met" if RSPUppRnd < Perf_NationRnd
replace RSP_NS = "Not met" if RSPLowRnd > Perf_NationRnd

* Count number of states meeting, not meeting, and no different from NS
egen RSP_Met=total(RSP_NS=="Met" & pickone)
egen RSP_NotMet=total(RSP_NS=="Not met" & pickone)
egen RSP_NotDif=total(RSP_NS=="No diff" & pickone)

* Count number of states that must engage in a PIP
gen RSP_Pip = ""
replace RSP_Pip = "No PIP" if RSP_NS == "Met"
replace RSP_Pip = "No PIP" if RSP_NS == "No diff"
replace RSP_Pip = "PIP" if RSP_NS == "Not met"

***************************
* REPORTING
***************************

* Export to excel the state's results, formatted for profile
gen Indicator_new = "Re-entry to foster care"
gen RSPRnd = round(RSP,0.001)
gen Perf_StateRnd = round(Perf_State,0.001)

gen RSP_Range = string(RSPLowRnd*100,"%02.1f") + "-" +
string(RSPUppRnd*100,"%02.1f")
gen RSP_Range_Formatted = string(RSPLowRnd*100,"%02.1f") + "%-" +
string(RSPUppRnd*100,"%02.1f") + "%"

gen Format = "Percent"
if substr(`"`CurrentCohort'"',1,2) ==  "AB"{
               gen Period = substr(`"`CurrentCohort'"',3,2) + "A" +
substr(`"`CurrentCohort'"',3,2) + "B"
                    gen DataUsed = substr(`"`CurrentCohort'"',3,2) + "A-"
+ string(real(substr(`"`CurrentCohort'"',3,2))+2) + "B"
}
else if substr(`"`CurrentCohort'"',1,2) == "BA" {
```

```
               gen Period = string(real(substr(`"`CurrentCohort'"',3,2))-
1) + "B" + substr(`"`CurrentCohort'"',3,2) + "A"
                    gen DataUsed =
string(real(substr(`"`CurrentCohort'"',3,2))-1) + "B-" +
string(real(substr(`"`CurrentCohort'"',3,2))+2) + "A"
}

* These variables only exist to ensure that there is both a column and
record with this data after reshaping below, and because tableau.
gen NS = string(`Perf_NationNS'*100,"%02.1f")+"%"
gen RSP_Range2 = RSP_Range_Formatted
gen RSP_NS2 = RSP_NS
gen DataUsed2 = DataUsed
gen MatchField = ""

outsheet stateabb statetxt Indicator_new NS TwelveMoCohort Period
DataUsed DataUsed2 Format RSP_NS RSP_NS2 RSP_Range RSP_Range_Formatted
RSP_Range2 RSP_Pip Den_State Num_State Perf_StateRnd RSPLowRnd RSPRnd
RSPUppRnd Perf_NationRnd MatchField ///
if (stateabb == `"`MyState'"' & pickone==1) using "`dirstub'/4 -
Performance modeled/RSP_Reentry_Summary `CurrentCohort' `MyState'.csv",
comma nolabel replace


import delimited "`dirstub'/4 - Performance modeled/RSP_Reentry_Summary
`CurrentCohort' `MyState'.csv", clear case(preserve)

rename (Den_State Num_State Perf_StateRnd RSPLowRnd RSPRnd RSPUppRnd
Perf_NationRnd) Result=
rename (RSP_Range2 RSP_NS2 DataUsed2) str=

reshape long Result str, i(stateabb statetxt) j(ValueType) string
replace Result = 0 if missing(Result)

rename (str) Result_String
rename (statetxt) state

outsheet stateabbstate Indicator_new    NS    TwelveMoCohort   Period
      DataUsed    Format     RSP_NS     RSP_Range   RSP_Range_Formatted
      RSP_Pip    ValueType   Result     Result_String MatchField ///
 using "`dirstub'/6 - Data for Tableau/Stata Output/RSP_Reentry_Summary
`CurrentCohort' `MyState'.csv", comma nolabel replace

export excel stateabb state  Indicator_new NS TwelveMoCohort Period
DataUsed Format RSP_NS RSP_Range RSP_Range_Formatted RSP_Pip ValueType
Result Result_String MatchField ///
 using "`dirstub'/6 - Data for Tableau/Stata Output/RSP_Reentry_Summary
`CurrentCohort' `MyState'.xlsx", nolabel replace firstrow (variables)


log close
}
}
```