EXAMPLE 8:
USING EGOALT FILES

DESCRIPTION: The objective of this example is to illustrate how to use **w_egoalt** files to (1) create cross-wave identifiers of family members such as spouse, partner, children, siblings, etc [This information can then be used to match the information of the linked household members as shown in Example 7]
(2) create a wide format file of own children living in the household
(3) identify changes in household composition, e.g, identifying leavers, stayers and joiners including identifying marital separations and child custody arrangements

FILES: **b_egoalt.dta**
WAVES: 2

NEW COMMANDS: **`inlist`, `preserve`, `restore`**

STEPS:

1. Open **b_egoalt** file and take a look at the data
2. Create unique identifiers for particular family members such as spouse or partner, father, mother, grandfather, grandmother.
3. Create a dataset consisting of cross-wave identifier, age and sex of respondent's co-resident children
4. Identify those who joined the respondent's household this wave, those who will not be living with the respondent in the next wave. Specifically, identify respondents who will experience marital separation next wave and those who will no longer be living with their children at the next wave

8.1 GETTING TO KNOW THE EGOALT FILE

The **w_egoalt** files are wave-specific derived datasets which use information from the household grid to show the relationship between any two household members. Every row of this dataset represents the relationship between any two household members. In order to be parsimonious with the relationship categories, these are expressed in gender neutral terms. The relationship variable combined with variables representing the sex of the two household members provides complete relationship information. The structure of the **w_egoalt** files is illustrated in Table 1. One of the household members is referred to as the EGO and the other as ALTER. The relationship variable, **w_relationship**, is expressed as the relationship of the EGO to the ALTER. In addition to this the dataset consists of the unique household identifier, **w_hidp**, the person numbers of the EGO and ALTER, **w_pno** and **w_apno**, the unique cross-wave personal identifiers of EGO and ALTER, **pidp** and **apidp**. For example, the first row of Table 1 shows that in household identified by **w_hidp** = 1001, the household member identified by **w_hidp** = 1001 & **w_pno** = 1 (or identified by **pidp = 100101**) is a woman who is living together with her partner, a man, identified by **w_hidp = 1001 & w_pno = 2** (or identified by **pidp =** 100102). They have an adopted daughter identified by **w_hidp** = 1001 & **w_pno** = 3 (or identified by **pidp** = 100103) and a biological daughter identified by **w_hidp =** 1001 & **w_pno** = 4 (or identified by **pidp =** 100104).

Table 1: Illustration of the structure of **w_egoalt** files

Household identifier	Person number of EGO	Person number of ALTER	pidp of EGO	pidp of ALTER	sex of EGO	sex of ALTER	Relationship of EGO to ALTER
w_hidp	**w_pno**	**w_apno**	**pidp**	**apidp**	**w_sex**	**w_asex**	**w_relationship_dv**
1001	1	2	100101	100102	Female	Male	partner/cohabittee
1001	1	3	100101	100103	Female	Female	adoptive parent
1001	1	4	100101	100104	Female	Female	natural parent
1001	2	1	100102	100101	Male	Female	partner/cohabittee
1001	2	3	100102	100103	Male	Female	adoptive parent
1001	2	4	100102	100104	Male	Female	natural parent
1001	3	1	100103	100101	Female	Female	adopted son/daughter
1001	3	2	100103	100102	Female	Male	adopted son/daughter
1001	3	4	100103	100104	Female	Female	adopted brother/sister
1001	4	1	100104	100101	Female	Female	natural son/daughter
1001	4	2	100104	100102	Female	Male	natural son/daughter
1001	4	3	100104	100103	Female	Female	adopted brother/sister

This file also consists of three other variables not shown in Table 1.

w_relationship : Some coding errors in **w_relationship** are corrected using information about the characteristics and data from other waves to create **w_relationship_dv**

w_elwstat : EGO's residence at the previous wave (not available in first wave)

w_alwstat : ALTER's residence at the previous wave (not available in first wave)

w_enwstat : EGO's residence at the next wave (not available in most recent wave)

w_anwstat : ALTER's residence at the next wave (not available in most recent wave)

The variables showing the status of the EGO and the ALTER in the previous and next waves are useful in identifying changes in household structure in a quick and easy manner.

Now open the dataset, **b_egoalt**, in Stata and take a look at it.

```
use "$dir/ukhls/b_egoalt", clear
```

An optional but useful step when using Understanding Society datasets is to drop the wave prefix. This generalizes your program across waves.

```
renpfix b_
```

Another optional but useful step is to save this dataset as a different file so that you do not accidentally over-write the main source file, **b_egoalt.dta**

```
save temp_egoalt, replace
```

By now you are familiar with the standard Stata commands for examining the data, use these to examine the dataset: **browse, list, summarize, describe, tabulate, fre.**

```
d
su
sort hidp pno apno
li in 1/10, sepby(hidp)
fre relationship_dv anwstat enwstat alwstat elwstat
tab relationship_dv sex
```

Or you could browse interactively.

Questions:

Which variable or variables uniquely identify each row?

How many men and women are living with their husband/wife, partner/ cohabitee or civil partner in this wave?

How many are living in same sex partnerships?

How many OSM children were born between last wave and this? How can you identify the new entrants in the dataset?

Digression

Note the **w_egoalt** is a file which shows the relationship between each household member with every other household member. So, single person households are excluded. To verify this we will compute the household size from **w_indall** and then compare the household size of matched and unmatched cases. Here we will demonstrate the usefulness of the Stata commands, `preserve` and `restore`. These allow you to temporarily make changes to the dataset and then restoring the original dataset once that is done. All commands between `preserve` and `restore` will work and produce the requested output but the original dataset will leave the original dataset unchanged. ***NOTE:*** `preserve, restore` ***and all the commands between them need to be run together. Otherwise the commands will not work correctly.***

```
preserve
use pidp b_hidp using "$dir/ukhls/b_indall", clear
bys b_hidp: g hhsize=_N
merge 1:m pidp using "$dir/ukhls/b_egoalt"
ta hhsize _merge
restore
```

8.2 CREATE UNIQUE IDENTIFIERS FOR SPECIFIC FAMILY MEMBERS

We will create a variable that records the unique cross-wave identifier of the EGO's spouse or partner and compare that with the spouse/partner identifier provided with the data in **w_indall**. To keep it simple, we will keep only those observations where the EGO is the husband/wife, partner/ cohabitee or civil partner of the ALTER. We will check if there are individuals with more than one partner and/or spouse because if there are then that will require different Stata commands to identify the multiple spouses or partners.

```
keep if inlist(relationship_dv,1,2,3)

bys pidp: g num_partners=_N
fre num_partners
```

We see that there are no individuals who are living in a household with two partners. As a general rule include syntax to make sure only non-duplicate cases are present,

```
keep if num_partners==1
```

Now we are ready to compute the unique cross-wave identifier for spouse or partner. As these identifiers have many digits it is necessary to specify the data type of the new variable such that the data type ("long") allows integers with up to 10 digits. The default is "float". For more information on this type,

```
help data type
```

New partner ID variable,

```
g long partner_pidp=apidp
```

To compare our variable with the identifiers provided in **b_indall** we will merge our data set with **b_indall**. Take a look at the online documentation (or Example 7) to find out which variable represents the unique cross-wave identifier of the spouse or partner: **w_ppid**

```
merge m:1 pidp using "$dir/ukhls/b_indall", keepusing(pidp b_ppid)
```

You will see that there are some individuals present in **b_indall** but not in our current dataset. This is because anyone who is not living with a spouse or partner is not in our current dataset but every individual in all enumerated households is in **b_indall**. Let us drop the unmatched cases.

```
drop if _m==2
```

Now let us count the cases when the partner ID we created is not equal to the one provided with the data.

```
count if partner_pidp != b_ppid
```

Make sure this number is ZERO. If not check your code.

Next we will create unique cross-wave identifiers of the father, mother, grandfather and grandmother of EGO. Comparison with identifiers provided with the data is left as an

exercise. Note that we had modified the **egoalt** dataset for creating partner IDs and so we cannot use that dataset. We will need to open the original dataset again.

```
use temp_egoalt, clear
```

One important difference between partner ID and IDs for father, mother, grandfather and grandmother is that we need to additionally take into account the gender of the ALTER.

```
g long mother_pidp=apidp ///
if relationship_dv>=4 & relationship_dv<=7 & asex==2

g long father_pidp=apidp ///
if relationship_dv>=4 & relationship_dv<=7 & asex==1

g long grandmother_pidp=apidp ///
if relationship_dv==20 & asex==2

g long grandfather_pidp=apidp ///
if relationship_dv==20 & asex==1
```

8.3 A (WIDE FORMAT) DATA SET OF CO-RESIDENT CHILDREN INCLUDING THEIR IDENTIFIER, AGE AND SEX

First we will restrict the dataset to the children of EGO,

```
use temp_egoalt, clear
keep if relationship_dv>=9 & relationship_dv<=12
```

Next we will attach ALTER's age variable to this dataset as it is not included in **egoalt**. To do this we will use a similar technique to the one we used in Example 7.

```
keep pidp apidp relationship_dv sex asex
rename pidp xpidp
rename apidp pidp
merge m:1 pidp using "$dir/ukhls/b_indall", ///
      keepusing(pidp b_dvage)
drop if _m==2
drop _m
```

Next, we will rename **pidp**, **b_dvage** and **asex** to variable names that signify that these are variables relating to the child.

```
rename pidp kpidp
lab var kpidp "cross-wave identifier of child"

rename b_dvage kage
lab var kage "child's age"

rename asex ksex
lab var ksex "child's sex"
```

Now, we can change the name of EGO's ID back to **pidp**.

```
rename xpidp pidp
```

Before we can convert this dataset of EGO's children into a wide format file we will need to create a variable that will identify the child number in the wide format file. Let us create that. While there is no reason to sort by child's age, by doing so, we create a datafile where the children are ordered with younger children having lower identifers, **k_id**

```
bys pidp (kage): g k_id =_n
```

Next convert it into a wide format file,

```
reshape wide kpidp ksex kage relationship_dv, i(pidp) j(k_id)
```

This dataset can be used for research on child outcomes by attaching additional variables (other than age and sex) relating to the child. We can use this simple dataset to look at the demographic characteristics of co-resident children. For example, what is the proportion of individuals living with at least one adult child? By taking a quick look at the data we see that the maximum number of children in the household is 11. Here we can again make use of loops to reduce the amount of Stata code we need to type which will reduce the likelihood of error. We have already used **foreach** loops. Here we will use another type of loop, the **forvalues** loop. The principle is the same, except that the latter is defined across numbers than items.

```
g adult_children_HH=0
forvalues i=1/11 {
      replace adult_children_HH=1 if kage`i'>=18 & kage`i'<.
}
fre adult_children_HH
```

8.4 IDENTIFYING JOINERS, LEAVERS

We can make use of the status variables to identify leavers and joiners:

elwstat: ego's residence at previous wave
alwstat: alter's residence at previous wave
enwstat: alter's residence at next wave
anwstat: ego's residence at next wave

Take a look at the value labels to understand what the numeric codes mean. For example, if we want to identify how many individuals joined a new household since last wave and how many will leave by next wave,

```
use temp_egoalt, clear

g JoinHH = 0
replace JoinHH = 1 if inlist(alwstat,2,3,4,5,6)
lab var JoinHH "Alter joined ego's HH this wave"

g LeftHH = 0
replace LeftHH = 1 if inlist(anwstat,2)
lab var LeftHH "Alter left ego's HH next wave"

fre JoinHH LeftHH
```

We can go further and identify marital separations and custody arrangements. First let us identify marital separations between this wave and the next:

```
generat separated=-9 if inlist(relationship_dv,1,2,3)
replace separated=0  if separated!=. & anwstat==1
replace separated=1  if separated!=. & anwstat==2 & enwstat==2
replace separated=2  if separated!=. & anwstat==3 & enwstat==2
replace separated=2  if separated!=. & anwstat==2 & enwstat==3
replace separated=3  if separated!=. & anwstat==6

lab var separated "Whether living with partner at next wave"
```

```
lab def separated -9 "missing"
lab def separated  0 "living together", modify
lab def separated  1 "separated", modify
lab def separated  2 "possibly separated", modify
lab def separated  3 "partner dead", modify
lab val separated separated

fre separated
```

Check why do we not have this information for all couples?

```
ta anwstat enwstat if separated==-9
```

Next we will identify whether children are living with their parents at next wave by creating a variable identifying future non-resident parents (**nrp**).

```
generat nrp=-9 if (relationship_dv>=9 & relationship_dv<=12)
replace nrp=0 if nrp!=. & anwstat==1
replace nrp=1 if nrp!=. & anwstat==2 & enwstat==2
replace nrp=2 if nrp!=. & anwstat==3 & enwstat==2
replace nrp=2 if nrp!=. & anwstat==2 & enwstat==3
replace nrp=3 if nrp!=. & anwstat==6

lab var nrp " Whether child not living in HH at next wave"

lab def nrp -9 "missing" ///
            0 "child in HH" ///
            1 "child not in HH" ///
            2 "child possibly not in HH" ///
            3 "child died"
lab val nrp nrp

fre nrp
ta anwstat enwstat if nrp==-9
```

Questions: How many individuals will separate by the next wave? How many of them are future non-resident parents?

Some clean up.

```
erase temp_egoalt.dta
```

ANSWERS

Questions: How many individuals will separate by the next wave? How many of them are future non-resident parents?
Answer: Use the following syntax to answer these questions

```
bys pidp: egen num_sep=total(separated==1|separated==2)
bys pidp: egen num_nrkids=total(nrp==1|nrp==2)
bys pidp: keep if _n==1
tab num_sep num_nrkids, m
```