

1) Ensuring that identifiers are unique

```
. by id: assert _N==1
```

With a "by id", `_N` refers to the total number of observations in the dataset that have that id number. Therefore, **by id: assert _N==1** asserts that, for each id number, there is one and only one observation. There are other ways you could verify that an id is really unique:

```
. duplicates report id
. isid id
```

If the id variable contained all unique values, **isid** would not give any output at all.

Review some merged situations:

Master dataset	Using dataset	Merged dataset
no identifiers	no identifiers	join observations
Uniq-id	Uniq-id	match-merge
non-u-id	Uniq-id	spread
Uniq-id	Non-u-id	spread
Non-u-id	Non-u-id	mistake

2) Merging Data: Individual-to-Community

Individual data: the data we already have for respondents (such as age, sex, race, education achievement, income...)

Community data: the characteristics of communities from which respondents sampled to the data (such as city, residential community (population, number of household, crime rate, average household income), school)

Suppose I have a dataset of students across universities. I would have one variable, `sid`, which identifies the students, as well as `uid` which identifies the university student from, and several individual level variables, such as sex, age. I also have a university level data which contains variables such as number of enrolled students (in thousand unit). Now I want to merge these two datasets together:

```
. list (student)
      +-----+
      | sid  uid  sex  age |
      +-----+
  1. | 101   1   m   20 |
  2. | 102   1   f   18 |
  3. | 103   1   f   25 |
  4. | 104   2   m   23 |
  5. | 105   3   f   21 |
      +-----+
  6. | 106   3   m   29 |
      +-----+
```

```

. list
+-----+
| uid  stud_num |
+-----+
1. | 1      21    |
2. | 2      43    |
3. | 3      35    |
+-----+

. use master,clear
. merge uid using second
. list

```

	sid	uid	sex	age	stud_num	_merge
1.	101	1	m	20	21	3
2.	102	1	f	18	21	3
3.	103	1	f	25	21	3
4.	104	2	m	23	43	3
5.	105	3	f	21	35	3
6.	106	3	m	29	35	3

Note: all students from same university have same number of enrolled students now

3). Updates

An update is a variation on a merge. In a formal merge the using data are used only to fill in the master around the edges (which is to say, the master-data part remains unchanged). In an update, information from the using dataset allowed to update the master-data part.

If I merge without using "update", let's look back at our result:

	id	a	b			id	a			id	a	b	_merge
1.	1001	10	11	+	1.	1001	10	=	1.	1001	10	11	3
2.	1002	12	13		2.	1003	14		2.	1002	12	13	1
3.	1003	.	15		3.	1005	20		3.	1003	.	15	3
4.	1005	16	17						4.	1005	16	17	3

Now I will use the word "update" to refer to making changes made inside the original master-data stub.

```

. use one,clear
. sort id
. merge id using two,update
. list
+-----+
| id  a  b  _merge |
+-----+
1. | 1001 10 11    3   |
2. | 1002 12 13    1   |
3. | 1003 14 15    4   |
4. | 1005 16 17    5   |
+-----+

```

+-----+

With that, here are what the codes in `_merge` mean:

- `_merge==1`. Observations came from the master dataset only, with no merge and no update.
- `_merge==2`. Observations came from the using dataset only, with no merge and no update.
- `_merge==3`. Observations were merged but not updated. Moreover, the updates in the using dataset, if there were any, were in agreement with what was already there.
- `_merge==4`. Observations were merged and updated. For example, at least one missing value in the master dataset was updated with a value from the using dataset.
- `_merge==5`. Observations were merged but not updated (the update was rejected). The result is the same as `_merge==3`, but there were updates in the using dataset that were inconsistent with what was already there, so the update was rejected.

Application of using `append` and `merge`:

`append` and `merge` are obviously useful in their own right, but I want to show you that even when you have a single dataset they are sometimes useful.

For instance, you could have a dataset containing information on married couples:

```
+-----+
|  fid  oh  h_age  h_inc  w_age  w_inc |
+-----+
1. | 101  0   34   3758   32   1217 |
2. | 102  0   24   3356   23   2412 |
3. | 103  1   45   5647   38   4213 |
4. | 104  1   56   8945   47   4561 |
5. | 105  0   31   5416   24   2243 |
+-----+
```

Question 1: I want to have individual data (one observation per person), how to do that?

```
/**Step 1: create husbands data***/
```

```
use family,clear
ren h_age age
ren h_inc income
drop w_age w_inc
gen hw=1
save husbands,replace
```

```
. list
```

```
+-----+
|  fid  oh  age  income  hw |
+-----+
1. | 101  0   34   3758   1 |
2. | 102  0   24   3356   1 |
3. | 103  1   45   5647   1 |
4. | 104  1   56   8945   1 |
5. | 105  0   31   5416   1 |
+-----+
```

```
/**Step 2: create wives data***/
```

```
use family,clear
```

```

ren w_age age
ren w_inc income
drop h_age h_inc
gen hw=0
save wives,replace
. list

```

	fid	oh	age	income	hw
1.	101	0	32	1217	0
2.	102	0	23	2412	0
3.	103	1	38	4213	0
4.	104	1	47	4561	0
5.	105	0	24	2243	0

/**Step 3: append husbands and wives data to form a new large individual dataset***/

```

use husbands,clear
sort fid
append using wives
. list

```

	fid	oh	age	income	hw
1.	101	0	34	3758	1
2.	102	0	24	3356	1
3.	103	1	45	5647	1
4.	104	1	56	8945	1
5.	105	0	31	5416	1
6.	101	0	32	1217	0
7.	102	0	23	2412	0
8.	103	1	38	4213	0
9.	104	1	47	4561	0
10.	105	0	24	2243	0

Question 2: Now, let's consider the reverse problem. Suppose now I have above data, and want to have single observations per couple instead, how to do that?

```

. /*****Step1:create husbands data*****/
. use individuals,clear
. keep if hw==1
. drop hw
. ren age h_age
. ren inc h_inc
. sort fid
. save husbands,replace
.
. /*****Step2:create wives data*****/
. use individuals,clear
. keep if hw==0
. drop hw
. ren age w_age
. ren inc w_inc
. sort fid
. save wives,replace
.

```

```

. /*****Step3: merge two individuals data to community data****/
. use husbands,clear
. sort fid
. merge fid using wives
. list

```

	fid	oh	h_age	h_inc	w_age	w_inc	_merge
1.	101	0	34	3758	32	1217	3
2.	102	0	24	3356	23	2412	3
3.	103	1	45	5647	38	4213	3
4.	104	1	56	8945	47	4561	3
5.	105	0	31	5416	24	2243	3

```

/*****
Collapsing data across observations
*****/

```

```

/**get the total income of each couple**/
use individuals,clear
list
collapse (sum) total_inc=income,by(fid)
list

```

```

/**get the average ageand income of each couple**/
use individuals,clear
list
collapse (mean) avg_inc=income avg_age=age,by(fid)
list

```

```

/**how to add these average variables to original data?*/
use individuals,clear
gen id=_n
order id
sort id fid
save,replace
list
collapse (mean) _inc=income avg_age=age,by(fid)
sort fid
save collapse,replace
list

```

```

use individuals,clear
sort fid
list
merge fid using collapse
list

```

```

/**Application 2: sale example**/

```

```

/**normal way to get the month total sale**/
use c:\sya6933\sale,clear
egen month1=sum(sale) in 1/31
egen month2=sum(sale) in 32/60

```

```

/**using collapse command**/
use c:\sya6933\sale,clear
collapse (sum) sum_mth=sale,by(month)
list

```