# How can I create dyad IDs in Stata?

Authored by

**stats writer**

July 1, 2024

Creating dyad IDs in Stata refers to the process of generating unique identifiers for pairs of individuals within a dataset. This is commonly used in social network analysis and other research fields to identify and analyze relationships between two individuals. To create dyad IDs in Stata, one can use the "egen" command and specify the variables to be used as identifiers, such as names or IDs of the two individuals. This will generate a new variable with a unique identifier for each dyad. Additionally, the "egen" command can also be used to create weighted dyad IDs based on the strength or frequency of interactions between individuals. Overall, creating dyad IDs in Stata allows for efficient and accurate analysis of relationships between pairs of individuals within a dataset.

## How can I create dyad IDs? | Stata FAQ

**Data management with dyadic data can be difficult. Before analyzing dyadic data, one may wish to know how many unique dyads appear in a dataset and create an ID variable at this dyad level. In this page, we will demonstrate how to create unique dyad IDs in Stata.**

**First, suppose individuals within teams rated each other on a skill. This data is collected in a dataset with a row for each rating given and the team, rater, and ratee are indicated. The same individuals act as both raters and ratees and each individual in the dataset has a unique ID. The**

**variable y is the**
**rating assigned to ratee by rater.**

**To create dyad IDs, we will first use integer values such that the maximum of**
**our ID variable is the number of unique dyads in the dataset. This approach will**
**work regardless of the format of the rater/ratee IDs. We will then**
**show a quicker method**
**for getting unique dyad IDs that requires that the rater/ratee IDs**
**be numeric.**

**use https://stats.idre.ucla.edu/stat/stata/faq/dyads, clear**
**list in 1/10**

**+-------------------------+**
**| team rater ratee y |**
**|-------------------------|**
**1. | 1 1 2 9 |**
**2. | 1 1 3 5 |**
**3. | 1 1 4 10 |**
**4. | 1 1 5 10 |**

**5. | 1 2 1 7 |**

**|-------------------------|**

**6. | 1 2 3 5 |**

**7. | 1 2 4 2 |**

**8. | 1 2 5 2 |**

**9. | 1 3 1 2 |**

**10. | 1 3 2 6 |**

**+-------------------------+**

### Approach 1

**We will use several egen commands to create dyad IDs. First, we will**
**create two variables using egen with the group option on team,rater, and ratee**
**in different orders.**
**egen with group takes a list of variables and orders and numbers the**
**combinations of the variable values. In this dataset, the individual rater IDs**
**only appear in one team, so we would see the same results running the egen**
**commands without team included. However, it is possible for raters**

**to be nested in teams where the same ID appears in multiple teams and should be**
**counted as different dyads. It is safer to add team than risk omitting it**
**when it is required.**

```
egen a = group(team rater ratee)
egen b = group(team ratee rater)
list in 1/10
```

```
+-----------------------------------+
| team rater ratee y a b |
|-----------------------------------|
1. | 1 1 2 9 1 5 |
2. | 1 1 3 5 2 9 |
3. | 1 1 4 10 3 13 |
4. | 1 1 5 10 4 17 |
5. | 1 2 1 7 5 1 |
|-----------------------------------|
6. | 1 2 3 5 6 10 |
7. | 1 2 4 2 7 14 |
8. | 1 2 5 2 8 18 |
9. | 1 3 1 2 9 2 |
10. | 1 3 2 6 10 6 |
```

+----------------------------------+

We can see that when we sort and number by rater and then ratee,
we get variable a. If we sort and number by ratee and then
rater, we get variable b. So a = 1 when rater = 1 and ratee = 2 and then b = 1 when ratee = 1 and rater = 2. Since we want to assign the same ID value to all observations involving
individuals 1 and 2 (regardless of their roles), we can see that all such
observations have the same pair of a and b values. We can take the
minimum of a and b to create a variable that uniquely identifies
dyads. We can then again use egen with group on the minimum to have a dyad ID
variable that does not skip integer values (as we can see will happen if we just
use the minimum as our ID variable).

```
egen c = rowmin(a b)
egen dyad = group(team c)
```

```
list in 1/10

+-----------------------------------------------+
| team rater ratee y a b c dyad |
|-----------------------------------------------|
1. | 1 1 2 9 1 5 1 1 |
2. | 1 1 3 5 2 9 2 2 |
3. | 1 1 4 10 3 13 3 3 |
4. | 1 1 5 10 4 17 4 4 |
5. | 1 2 1 7 5 1 1 1 |
|-----------------------------------------------|
6. | 1 2 3 5 6 10 6 5 |
7. | 1 2 4 2 7 14 7 6 |
8. | 1 2 5 2 8 18 8 7 |
9. | 1 3 1 2 9 2 2 2 |
10. | 1 3 2 6 10 6 6 5 |
+-----------------------------------------------+
```

To see how many unique dyads appear in your data, you can use the codebook
command to see the number of unique values.

```
codebook(dyad)
```

---

---

dyad dayd id

---

---

type: numeric (float)

range: units: 1
unique values: 500 missing .: 0/1000

mean: 250.5
std. dev: 144.41

percentiles: 10% 25% 50% 75% 90%
50.5 125.5 250.5 375.5 450.5

### Approach 2

Alternatively, we could have created unique dyad IDs in one step.

```
gen dyad2 = 1000*team+100*(rater + ratee) + min(rater, ratee)
list team rater ratee dyad2 in 1/10
```

```
+-----------------------------+
| team rater ratee dyad2 |
|-----------------------------|
1. | 1 1 2 1301 |
2. | 1 1 3 1401 |
3. | 1 1 4 1501 |
4. | 1 1 5 1601 |
5. | 1 2 1 1301 |
|-----------------------------|
6. | 1 2 3 1502 |
7. | 1 2 4 1602 |
8. | 1 2 5 1702 |
9. | 1 3 1 1401 |
10. | 1 3 2 1502 |
+-----------------------------+
```

**These IDs will no longer be the integer values starting at 1 and going up to the number of unique dyads in the data, but we can see we have the same number of unique values.**

**codebook dyad2**

---

---

**dyad2 (unlabeled)**

---

---

**type: numeric (float)**

**range: units: 1**
**unique values: 500 missing .: 0/1000**

**mean: 80271.5**
**std. dev: 46361**

**percentiles: 10% 25% 50% 75% 90%**
**16052 40133.5 80272 120409 144492**