Title

stata.com

Intro 4 — Preparing data: Data with shapefiles

Description Remarks and examples Also see

Description

To perform a spatial analysis, you do the following steps:

1. Prepare data for use by Sp.
2. Define weighting matrices.
3. Fit models using `spregress`, `spivregress`, or `spxtregress`.

Step-by-step instructions for step 1 are provided below. These instructions are for preparing data with shapefiles.

Shapefiles define maps. You obtain them over the web. After translation into Sp format, you merge the translated result with a `.dta` file or files you already have.

You may also be interested in introductions to other aspects of Sp. Below, we provide links to those other introductions.

Intro 1	A brief introduction to SAR models
Intro 2	The **W** matrix
Intro 3	Preparing data for analysis
Intro 5	Preparing data: Data containing locations (no shapefiles)
Intro 6	Preparing data: Data without shapefiles or locations
Intro 7	Example from start to finish
Intro 8	The Sp estimation commands

Remarks and examples

stata.com

Remarks are presented under the following headings:

Overview

Shapefile is jargon for computer files that store a map. A shapefile might store the map for the 3,000-plus counties in the United States.

Shapefiles are optional. If you have one, Sp can determine which places (counties) are neighbors (share a border), and Sp will know the distances between the centroids of the places. You will be able to create spatial weighting matrices of first-order neighbors by typing

```
. spmatrix create contiguity Wc
```

and to create inverse-distance weighting matrices by typing

```
. spmatrix create idistance Wd
```

and to graph choropleth maps by typing

```
. grmap ue_rate
```

You find and download shapefiles on the web, and translate them to Stata format. For example,

1. You find and download `tl_2016_us_county.zip` for U.S. counties.
2. You convert `tl_2016_us_county.zip` to Stata format, creating two new datasets: `tl_2016_us_county.dta` and `tl_2016_us_county_shp.dta`.

For information on how to find `tl_2016_us_county.zip` on the web, see *Finding a shapefile for Texas counties* in [SP] **Intro 7**. You can download this file if you want to follow along with the commands in this introduction.

Let's suppose you have downloaded the U.S. counties file and unzipped it. You also have two county-data datasets, `project_cs.dta` and `project_panel.dta`, containing observations on the 3,000-plus counties. These datasets are available by typing

```
. copy https://www.stata-press.com/data/r17/project_cs.dta .
. copy https://www.stata-press.com/data/r17/project_panel.dta .
```

They are standard Stata datasets. You could use them with `regress` or, in the case of `project_panel.dta`, which contains panel data, `xtreg`, but the datasets are not yet suitable for use with `spregress` or `spxtregress`.

To make the project datasets work with Sp, you merge each one with the Stata-format shapefiles. We will show you how to create these shape files in *Creating Stata-format shapefiles*. Merging your data with a shapefile will add three variables to your data: `_ID`, `_CX`, and `_CY`.

`project_cs.dta` is a cross-sectional dataset, so when the shapefile is prepared, you could type

```
. use project_cs, clear
. merge 1:1 fips using tl_2016_us_county
. keep if merge==3
. drop _merge
```

If all goes well, no observations from `project_cs.dta` will be dropped. You keep the matches because there are sometimes observations in the shapefile that are not in `project_cs.dta`.

`project_panel.dta` is a panel dataset, so you could type

```
. use project_panel, clear
. xtset fips time
. spbalance
. merge m:1 fips using tl_2016_us_county
. keep if _merge==3
. drop _merge
```

Merging panel data requires extra steps because 1) the data must be `xtset` and 2) Sp requires that the panels be strongly balanced. This was discussed in [SP] **Intro 3**.

How to find and download shapefiles on the web

Shapefiles contain more than a map. They sometimes contain data relevant to specific research problems. You can find shapefiles that contain climate or economic or epidemiological data. You might think that you need to find the shapefile relevant to your research problem, but you do not. You need to find shapefiles defining the geographic units that you will be analyzing, such as U.S. counties. In addition to the map, shapefiles include the names and standard codes for the geographic units. You will later use those variables to merge the shapefile with data you already have or that you obtain from other sources.

To find appropriate shapefiles, use your browser and search for them. You could search for

```
shapefiles

shapefiles europe
shapefiles deutschland
shapefiles deutschland bundesländer
shapefiles schweiz kantone
shapefiles uk
shapefiles uk county

shapefiles us
shapefiles us census
shapefiles us census county
shapefiles us census blocks

shapefiles us census tiger     // TIGER/Line is especially popular
```

It is best to choose a shapefile from official sources. If such a shapefile is not available, choose one that is from a reputable source.

Find the appropriate shapefile and download it.

Standard-format shapefiles

The shapefile you just loaded is known as a standard-format shapefile. The word shapefile itself is confusing because a shapefile is actually a collection of related files. For example, a shapefile could be any of the following:

File	Contents
name`.shp`	shapes and locations of geographic units
name`.dbf`	other attributes of the geographic units
name`.*`	other information, not needed by Sp
name`.zip`	compressed file containing all the files above

name`.zip` is often called a shapefile even though it is a zip file containing the shapefiles.

name`.shp` really is a shapefile—it contains the map of the geographic units, which could be countries of the world, counties of the United States, etc.

name`.dbf` contains data (called attributes) about the geographic units. The `.dbf` stands for database file. It is a dataset containing variables and observations. Among the variables are usually variables for the names and numeric identification codes of the geographic units. The file sometimes contains other variables, such as temperature, rainfall, or unemployment. After translation to Sp format, you can use the variables, ignore them, or drop them.

In addition to *name*`.shp` and *name*`.dbf`, there are other files. Stata ignores them, and you can erase them if you wish. After translation, you can erase all the files that were in the original `.zip` file.

Stata-format shapefiles

You will translate the standard-format shapefiles to Stata format. It is easy to do:

```
. unzipfile name.zip
. spshape2dta name
```

This will create two Stata-format datasets, *name*`.dta` and *name*`_shp.dta`.

Stata-format file	Corresponding standard-format file
name`.dta`	*name*`.dbf`
name`_shp.dta`	*name*`.shp`

name`.dta` contains

Variable name	Contents
`_ID`	ID variable with values $1, 2, \ldots, N$
`_CX`	x coordinate of centroid of geographic unit
`_CY`	y coordinate of centroid of geographic unit
other variables	attributes of the geographic units from *name*`.dbf`

Notes:
1. The dataset will have N observations, one for each geographic unit.
2. You will learn later that Sp data must be `spset`. `spshape2dta` handles that for you. *name*`.dta` is `spset`.
3. Variable `_ID` links observations in *name*`.dta` with the map stored in *name*`_shp.dta`.
4. You may rename, modify, or drop any of the variables except `_ID`, `_CX`, and `_CY`.
5. You merge your `.dta` files with *name*`.dta` to use them in Sp.

name`_shp.dta` contains

Variable name	Contents
`_ID`	ID variable with values $1, 2, \ldots, N$
other variables	descriptions of the map

Notes:
1. This file has many more than N observations. Each observation describes a line segment that when combined draws the map.
2. You do not use or modify this dataset. Sp uses the dataset behind the scenes.
3. *name*`.dta` and *name*`_shp.dta` must be in the same directory.

Creating Stata-format shapefiles

There are six steps to preparing shapefiles for use:

1. Find and download a standard-format shapefile.
2. Translate the shapefile to Stata format.
3. Look at the translated data.
4. Create a common ID variable for use with other data.
5. Optionally, tell Sp to use the common ID variable.
6. Set the units of the coordinates, if necessary.

These steps are not independent; that is, you cannot jump ahead to, say, step 4.

Below, we start at step 1, finding and downloading

```
tl_2016_us_county.zip
```

and finish with step 6, having created

```
tl_2016_us_county.dta
tl_2016_us_county_shp.dta
```

These are the same files we used in *Overview*.

We discuss each step below. Here is a preview of the code for the steps:

Step 1: Find and download a standard-format shapefile

```
. * do this on the web
```

Step 2: Translate the shapefile to Stata format

```
. copy ~/Downloads/tl_2016_us_county.zip .
. unzipfile tl_2016_us_county.zip
. spshape2dta tl_2016_us_county
```

Step 3: Look at the translated data

```
. use tl_2016_us_county, clear
. describe
. list in 1/5
```

Step 4: Create a common ID variable for use with other data

```
. generate long fips = real(STATEFP + COUNTYFP)
. bysort fips: assert _N==1
. assert fips != .
```

Step 5: Optionally, tell Sp to use the common ID variable

```
. spset fips, modify replace
```

Step 6: Set the units of the coordinates, if necessary

```
. spset, modify coordsys(latlong, miles)
. save, replace
```

Step 1: Find and download a shapefile

Use your browser. We did, and we found and downloaded `tl_2016_us_county.zip` as described in *Finding a shapefile for Texas counties* in [SP] **Intro 7**. Our browser stored the file in our `Downloads` directory, which is `~/Downloads/` on our computer. `~` is Stata syntax for home directory.

Step 2: Translate the shapefile to Stata format

We entered Stata and changed to the directory containing the project datasets. We typed

```
. copy ~/Downloads/tl_2016_us_county.zip .
. unzipfile tl_2016_us_county.zip
    inflating: tl_2016_us_county.cpg
    inflating: tl_2016_us_county.dbf
    inflating: tl_2016_us_county.prj
    inflating: tl_2016_us_county.shp
    inflating: tl_2016_us_county.shp.ea.iso.xml
    inflating: tl_2016_us_county.shp.iso.xml
    inflating: tl_2016_us_county.shp.xml
    inflating: tl_2016_us_county.shx
successfully unzipped tl_2016_us_county.zip to current directory
. spshape2dta tl_2016_us_county
  (importing .shp file)
  (importing .dbf file)
  (creating _ID spatial-unit id)
  (creating _CX coordinate)
  (creating _CY coordinate)
  file tl_2016_us_county_shp.dta created
  file tl_2016_us_county.dta     created
```

`spshape2dta` translated the files to Stata format. It did not load them into memory. You will never load the `*_shp.dta` file, but Sp will use it behind the scenes. The file is linked to `tl_2016_us_county.dta`, which you will directly use. Keep them both in the same directory.

Step 3: Look at the translated data

Look at the data you have just created. The data are already `spset`, but we can type `spset` to find out how:

```
. use tl_2016_us_county, clear
. spset
      Sp dataset: tl_2016_us_county.dta
Linked shapefile: tl_2016_us_county_shp.dta
            Data: Cross sectional
 Spatial-unit ID: _ID
     Coordinates: _CX, _CY (planar)
```

Look at the variables, too:

```
. describe
  (output omitted)
. list in 1/5
  (output omitted)
```

You need to understand the data and its variables. Some of them you will not need. You may drop them, but do not drop `_ID`, `_CX`, and `_CY`. They were created by `spshape2dta`, and you will need them later.

In the unlikely event that you find all the variables you need for your intended analysis, you can use tl_2016_us_county.dta as your analysis dataset. You are ready to go, except you might need to set the coordinate system. Skip to step 6, and stop after that.

Step 4: Create a common ID variable for use with other data

We continue with step 4 because we did not find the analysis variables we needed, nor did we expect to find them. We have project_cs.dta containing our analysis variables. The problem is that we will need to merge project_cs.dta with the Stata-format shapefiles, and to do that, they will need to have an ID variable in common. project_cs.dta has a variable named fips containing standard county codes. We hope to find the same variable in tl_2016_us_county.dta.

We looked but did not find the FIPS-code variable. We did discover the variable NAME containing county names. That variable could work for us. project_cs.dta also has a variable named countyname. If we rename NAME to countyname in tl_2016_us_county.dta, we could merge datasets.

However, we have had bad experiences merging on string variables. Names in the two datasets can differ for trivial reasons, such as capitalization. Before we resigned ourselves to the string-variable solution, we looked again. Numeric ID variables are better.

We discovered variables STATEFP and COUNTYFP. They were recorded as string variables, but appeared to contain two- and three-digit numeric codes. We read about FIPS codes on the web and learned there are two-digit state codes, three-digit county-within-state codes, and five-digit county codes, which are nothing more than the two- and three-digit codes run together. If STATEFP is 01 and COUNTYFP is 001, then the five-digit code is 01001.

We create the new numeric variable fips containing the run-together code by typing

```
. generate long fips = real(STATEFP + COUNTYFP)
```

The variable we created did not have to be numeric, but fips is numeric in project_cs.dta, and numeric is better for reasons to be explained in step 5.

In any case, we were pleased when we listed the value of variable NAME for fips = 1001 and it was Autauga.

We also verify that new variable fips really does uniquely identify the observations in tl_2016_us_county.dta by typing

```
. bysort fips: assert _N==1
. assert fips != .
```

Step 5: Optionally, tell Sp to use the common ID variable

This step is optional but worth doing if you found or created a numeric ID variable in the previous step. Because we created fips in step 4, we will type

```
. spset fips, modify replace
  (_shp.dta file saved)
  (data in memory saved)

      Sp dataset: tl_2016_us_county.dta
Linked shapefile: tl_2016_us_county_shp.dta
            Data: Cross sectional
 Spatial-unit ID: _ID (equal to fips)
     Coordinates: _CX, _CY (planar)
```

The above resets _ID. spset verifies that fips is numeric and would make an appropriate ID code. If it does, spset copies fips to Sp's _ID variable, the variable that officially identifies the observations. Sp then reindexes both tl_2016_us_county.dta and tl_2016_us_county_shp.dta on the new _ID values.

You should do this step because, if _ID is a common code, the spatial weighting matrices you create will be sharable with other projects and researchers. The rows and columns of the matrices will be identified by the common code rather than the arbitrary code _ID previously contained.

Step 6: Set the units of the coordinates, if necessary

The coordinates recorded in shapefiles historically were required to be in planar units. These days, shapefiles are just as likely to contain latitude and longitude. Usage is running ahead of file-format standards, and so you must determine which coordinate system is being used.

When Sp converts a shapefile as we did in step 2, it assumes coordinates are in planar units. If they are actually recorded in degrees latitude and longitude, you need to type

```
. spset, modify coordsys(latlong, miles)
```

or

```
. spset, modify coordsys(latlong, kilometers)
```

Whether you specify miles or kilometers is of little importance—that setting merely determines the units in which Sp will report distances. It is important, however, that you specify the coordinate system is latlong when it is latitude and longitude if distances are to be measured accurately.

The distributor of the shapefile may provide documentation that tells you whether the file uses planar units or latitude and longitude. If you are unable to find this information, you can do some detective work to figure it out.

Here is how to determine the units. Coordinates (centroids) are stored in variables _CX and _CY. We listed some of them and discovered that Brazos County, Texas, is recorded as being at

$$\_CX = -96.302386 \quad \text{and} \quad \_CY = 30.6608$$

We looked on the web and found that College Station, a city in Brazos County, is located at latitude 30.601389 and longitude −96.314444. We checked two other cities and counties and found similar agreement. (Note that latitude is stored in _CY and longitude in _CX. It will always be that way.)

Thus, we type

```
. spset, modify coordsys(latlong, miles)

      Sp dataset: tl_2016_us_county.dta
  Linked shapefile: tl_2016_us_county_shp.dta
             Data: Cross sectional
 Spatial-unit ID: _ID (equal to fips)
     Coordinates: _CY, _CX (latitude-and-longitude, miles)
```

We are finished preparing our shapefile, so we save `tl_2016_us_county.dta`.

```
. save, replace
file tl_2016_us_county.dta saved
```

Preparing your data

We now have

```
tl_2016_us_county.dta
tl_2016_us_county_shp.dta
```

These are the same datasets we used in *Overview*.

You should keep these two files around, just as they are. You can use them in the future whenever you have a county dataset that you want to use with Sp.

Step 7a: Merge your cross-sectional data with the Stata-format shapefiles

We showed you how to do this in the *Overview*, but we will do it again now that we have our Stata-format shapefiles so that you can see the output. To make the cross-sectional data in `project_cs.dta` work with Sp, type

```
. use project_cs, clear
. merge 1:1 fips using tl_2016_us_county
. keep if _merge==3
. drop _merge
. save, replace
```

The result is

```
. use project_cs, clear
(My cross-sectional data)
. merge 1:1 fips using tl_2016_us_county
    Result                      Number of obs
    -----------------------------------------
    Not matched                            91
        from master                         0  (_merge==1)
        from using                         91  (_merge==2)
    Matched                             3,142  (_merge==3)
    -----------------------------------------

. keep if _merge==3
(91 observations deleted)
. drop _merge
. save, replace
file project_cs.dta saved
```

Note that all observations from the master were matched. Had observations been dropped from the master, we would have found out why `project_cs.dta` contained counties not in `tl_2016_us_county.dta`.

We have not discussed the `spset` command, the other way to turn regular Stata datasets into Sp datasets. We will discuss `spset` in [SP] **Intro 5** and [SP] **Intro 6**. Merging regular data (`project_cs.dta`) with `spset` data (`tl_2016_us_county.dta`, because it was created by `spshape2dta`) produces an `spset` result. `project_cs.dta` was not `spset` before the merge, but it is now:

```
. spset
      Sp dataset: project_cs.dta
Linked shapefile: tl_2016_us_county_shp.dta
            Data: Cross sectional
 Spatial-unit ID: _ID (equal to fips)
     Coordinates: _CY, _CX (latitude-and-longitude, miles)
```

Step 7b: Merge your panel data with the Stata-format shapefiles

Because `project_panel.dta` is panel data, you still merge with `tl_2016_us_county.dta`, but you go about it a little differently. You type

```
. use project_panel, clear
. xtset fips time
. spbalance
. merge m:1 fips using tl_2016_us_county
. keep if _merge==3
. drop _merge
. save, replace
```

The result is

```
. use project_panel, clear
(My panel data)
. xtset fips time
Panel variable: fips (strongly balanced)
 Time variable: time, 1 to 3
         Delta: 1 unit
. spbalance
  (data strongly balanced)
. merge m:1 fips using tl_2016_us_county
    Result                      Number of obs
    -----------------------------------------
    Not matched                            91
        from master                         0  (_merge==1)
        from using                         91  (_merge==2)
    Matched                             9,426  (_merge==3)
    -----------------------------------------
. keep if _merge==3
(91 observations deleted)
. drop _merge
. save, replace
file project_panel.dta saved
```

project_panel.dta is now spset:

```
. spset
      Sp dataset: project_panel.dta
Linked shapefile: tl_2016_us_county_shp.dta
            Data: Panel
 Spatial-unit ID: _ID (equal to fips)
         Time ID: time (see xtset)
     Coordinates: _CY, _CX (latitude-and-longitude, miles)
```

The data are still xtset, but Sp modified the setting. The data were set on fips and time. They are now set on _ID and time:

```
. xtset
Panel variable: _ID (strongly balanced)
 Time variable: time, 1 to 3
         Delta: 1 unit
```

Sp changed the setting because spset and xtset must agree on the panel identifier.

Rules for working with Sp data, whether cross-sectional or panel

The data whether cross-sectional, as in project_cs.dta, or panel, as in project_panel.dta, is now Sp. It is a Stata dataset with one special feature: its observations are linked to the Stata-format shapefile tl_2016_us_shp.dta. Because of the linkage, there are rules for using either project_cs.dta or project_panel.dta.

Rule 1: Do not drop or modify variables _ID, _CX, or _CY.

You may drop other variables in the file.

Rule 2:

Cross-sectional data:

Do not add new observations.

Panel data:

Do not add new observations with new values of _ID.

The rule that handles both cross-sectional and panel data is that you may not add observations that have no corresponding definition in tl_2016_us_shp.dta.

For cross-sectional data, the rule reduces to "do not add new observations".

For panel data, the rule said positively is that you can add new observations, but only for new time periods within panels.

You may drop observations from cross-sectional data, and observations for entire panels from panel data. Dropping is allowed because unnecessary definitions in tl_2016_us_shp.dta are ignored.

Be careful when performing merges with other datasets. If you type

Cross-sectional data:

```
. merge 1:1 fips using anotherdataset
```

Panel data:

```
. merge 1:1 fips time using anotherdataset
```

or

```
. merge m:1 fips using anotherdataset
```

you must then either

```
. keep if _merge==3
```

or

```
. keep if _merge==1
```

Rule 3: Do not erase, modify, or rename file tl_2016_us_shp.dta.

Even if you rename `project_cs.dta` or `project_panel.dta`, do not rename `tl_2016_us_shp.dta`.

Rule 4: project_cs.dta or project_panel.dta and tl_2016_us_shp.dta must be stored in the same directory.

If you copy `project_cs.dta` or `project_panel.dta` to a different directory, copy `tl_2016_us_shp.dta` to the same directory.

That is the end of the prohibitions. The following rule need not be stated, because that which is not prohibited is allowed, but it is reassuring:

Rule 5: You may save copies of project_cs.dta or project_panel.dta under new names.

New files will inherit the linkage to `tl_2016_us_shp.dta`. For example, you could type

```
. copy project_cs.dta newname.dta
```

Afterward, if you wished, you could type

```
. erase project_cs.dta
```

Here is one way making copies can be useful:

```
. use project_cs
. keep if state=="Texas"
. save texas
```

Also see

[SP] **Intro 7** — Example from start to finish

[SP] **spset** — Declare data to be Sp spatial data

[SP] **spshape2dta** — Translate shapefile to Stata format