Title

stata.com

spset — Declare data to be Sp spatial data

Description

Data must be `spset` before you can use the other Sp commands. The `spset` command serves three purposes:

1. It reports whether the data are `spset` and if so, how.
2. It sets the spatial data for the first time.
3. It modifies how the data are `spset` at any time.

Data that are `spset` are called Sp data.

Quick start

Query whether or how data are `spset`

```
spset
```

In cross-sectional data, specify geographic unit identifier `id`

```
spset id
```

Add coordinates stored in variables `x` and `y` to previously `spset` data

```
spset, modify coord(x y)
```

In panel data, specify geographic unit identifier `id` and time within area identifier `tvar`

```
xtset id tvar
spset id
```

Menu

Statistics > Spatial autoregressive models

Syntax

Display the current setting

```
spset
```

Set data with shapefiles

```
spshape2dta ...
```
(see [SP] **spshape2dta**)

Set data without shapefiles

```
spset idvar [ , options ]
```

Modify how data are set with shapefiles

```
spset [idvar], modify [shpmodoptions]
```

Modify how data are set without shapefiles

```
spset [idvar], modify [modoptions]
```

Clear the setting

```
spset, clear
```

idvar is an existing, numeric variable that uniquely identifies the geographic units, meaning the observations in cross-sectional data and the panels in panel data.

shapefile refers to a Stata-format shapefile, specified with or without the `.dta` suffix. Such files usually have names of the form *name*`_shp.dta`.

options	Description
`coord(`*xvar yvar*`)`	designate *xvar* and *yvar* as the location coordinates
`coordsys(`*coordsys*`)`	specify how coordinates are interpreted

shpmodoptions	Description
`coordsys(`*coordsys*`)`	change how coordinates are interpreted
`noshpfile`	break link with shapefile
`replace`	replace current geographic identifier with *idvar*

modoptions	Description
`coord(`*xvar yvar*`)`	replace location coordinates with *xvar* and *yvar*
`coordsys(`*coordsys*`)`	change how coordinates are interpreted
`nocoord`	clear coordinate settings
`shpfile(`*shapefile*`)`	establish link to shapefile

Options

`coord(`*xvar yvar*`)` and `nocoord` specify coordinates. `coord()` specifies the variables recording the x and y coordinates or the longitude and latitude. `nocoord` specifies that previously set coordinates be forgotten.

`coord(`*xvar yvar*`)` creates or replaces the contents of Sp variables `_CX` and `_CY`.

`coord()` and `nocoord` are allowed only if the data are not linked to a shapefile. If you want to use different coordinates than the shapefile provides, break the connection to the shapefile by typing

```
. spset, modify noshpfile
```

and then use `spset, modify coord(`*xvar yvar*`)`. You can later use `spset, modify shpfile(`*shapefile*`)` to reestablish the link. Relinking to the shapefiles reestablishes the original coordinates stored in `_CX` and `_CY`.

`coordsys(`*coordsys*`)` specifies how to interpret coordinates. You may specify `coordsys()` regardless of whether you are linked to a shapefile. `coordsys()` syntax is

`coordsys(planar)`	(default)
`coordsys(latlong)`	(kilometers implied)
`coordsys(latlong, kilometers)`	
`coordsys(latlong, miles)`	

`coordsys(latlong)` specifies latitude and longitude coordinates. `kilometers` and `miles` specify the units in which distances should be calculated. Distances for `planar` coordinates are always in the units of the planar coordinates.

`modify` specifies that existing `spset` settings are to be modified. Omitting `modify` means that the data are being `spset` for the first time.

You can modify Sp settings as often as you wish.

`clear` clears all Sp settings. It drops the variables `_ID`, `_CX`, and `_CY` that `spset` previously created.

`noshpfile` breaks the link to the Stata-format shapefile, the file that usually has *shapefile*`_shp.dta`. Data that were linked to a shapefile will be just as if they had never been linked to it. Before breaking the link, you should make a note of the shapefile's name:

```
. spset            (make a note of the shapefile's name)
. spset, modify noshpfile
```

The shapefile might have been named *shapefile*`_shp.dta`. You will need the name later should you wish to reestablish the link.

`shpfile(`*shapefile*`)` and `drop` are for linking or relinking to a shapefile. To reestablish the link to the shapefile that was just unlinked above, you would type

```
. spset, modify shpfile(shapefile_shp)
```

Not only will the shapefile be relinked, but the coordinates stored in `_CX` and `_CY` will be restored, too.

`shpfile()` will refuse to link the shapefile if the data in memory contain observations for `_ID` values not found in the shapefile. In this case, specify `shpfile()` and `drop` if you are willing to drop the extra observations from the data in memory.

Remarks and examples

stata.com

Remarks are presented under the following headings:

Determining whether and how data are spset

`spset` without arguments queries the Sp setting. Data starts out not being `spset`:

```
. spset
data not spset
r(459);
```

After the data have been `spset`, the output might be

```
. spset
  Sp dataset
                data:  cross sectional
   spatial-unit ID:  _ID
       coordinates:  _CY, _CX (latitude-and-longitude, miles)
  linked shapefile:  shapefile_shp.dta
```

These data are as described in [SP] **intro 4**. They are linked to a Stata-format shapefile.

Or, the output might be

```
. spset
  Sp dataset
                data:  cross sectional
   spatial-unit ID:  _ID (equal to fips)
       coordinates:  _CY, _CX (latitude-and-longitude, miles)
  linked shapefile:  none
```

These data are as described in [SP] **intro 5**. The data contain coordinates but are not linked to a shapefile.

Or, the output might be

```
. spset
  Sp dataset
                data:  cross sectional
   spatial-unit ID:  _ID (equal to fips)
       coordinates:  none
  linked shapefile:  none
```

These data are as described in [SP] **intro 6**. They do not contain coordinates nor are they linked to a shapefile.

All the examples above are for cross-sectional data. If the data were panel data, the output might be

```
. spset
  Sp dataset
                  data:  panel
     spatial-unit ID:  _ID
             time id:  time (see xtset)
          coordinates:  _CY, _CX (latitude-and-longitude, miles)
    linked shapefile:  shapefile_shp.dta
```

Setting data for the first time

There are two kinds of data as far as Sp is concerned: cross-sectional and panel. In brief, cross-sectional data contain one observation per spatial unit, such as one observation per county. Panel data contain multiple observations, such as one observation per county per calendar year. The kinds of data are described in more detail in [SP] **intro 3**.

We are about to explain the various `spset` cases one at a time. We will discuss cross-sectional and panel data together. In all the examples, we will assume that you want to `spset` *analysis*`.dta`. This example dataset has the following characteristics:

1. It is cross-sectional or panel.
2. It contains data on U.S. counties. Variable `fips` contains the standard federal information processing standard (FIPS) code identifying U.S. counties.

If the data are cross-sectional, then `fips` uniquely identifies the observations.

If the data are panel, then variable `time` will be assumed to contain the second-level identifier. `fips` and `time` uniquely identify the observations. The time variable need not be named `time`, nor is the second-level identifier required to be time. See [SP] **intro 3**.

`spset` adds one or three variables to your data.

1. `_ID`, which identifies the geographical areas.
2. `_CX` and `_CY`, which record the coordinates of the areas. Variables `_CX` and `_CY` are added only if the coordinates are known.

`spset` also adds information stored in Stata characteristics.

3. *coordsys*, the system in which coordinates are recorded and whether distances should be measured in kilometers or miles.
4. *shpfile*, the name of the Stata-format shapefile to which the data are linked, if they are linked.

The variables and characteristics that `spset` adds to your data should be viewed as `spset`'s property. Do not modify or drop them. Use `spset, modify` to change settings.

Setting data with a standard-format shapefile

Shapefiles contain maps for each of the spatial units, which we will imagine are counties of the United States. You obtain shapefiles over the web.

You use [SP] **spshape2dta** to translate standard-format `*.zip` shapefiles to Stata-format `*_shp.dta` files. How you do that is explained in [SP] **intro 4**.

`spshape2dta` performs the initial Sp setting of the data for you. That initial setting will be

```
. spset
    data:  cross sectional
 spatial-unit id:  _ID
     coordinates:  _CY, _CX (planar)
linked shapefile:  shapefile_shp.dta
```

Note that spshape2dta derived the centroid coordinates for each of the spatial units (counties) and spset them.

You can modify settings. One important setting specifies how the coordinates are recorded. They are either planar, which is another word for rectangular, or they are degrees latitude and longitude. By default, Sp assumes coordinates are planar. Sp provides two coordinate-system settings:

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

It is important that you modify coordsys() to be latlong if that is what the data record, because the formulas for calculating distances differ; see [SP] **spdistance**. Sp datasets record the coordinate values in variables _CX and _CY.

coordsys(latlong) has an extra setting that may be important to you:

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

By default, coordsys(latlong) calculates distances in kilometers.

Setting data with a Stata-format shapefile

All shapefiles start out as standard-format shapefiles and are translated into Stata-format _shp.dta files. It is possible that you have a Stata-format _shp.dta file from a previous analysis that is appropriate for this analysis. In that case, you can just link to it.

Let's assume that we want to spset *analysis*.dta, which you may recall is county data and contains variable fips (and time if it is panel data).

Let's assume that you also have Stata-format shapefile *shapefile*_shp.dta from a previous analysis. The _shp.dta file is indexed on FIPS codes.

To spset the data and link them to *shapefile*_shp.dta, type

Cross-sectional data:

```
. use analysis
. spset fips
. spset, modify shpfile(shapefile_shp)
```

Panel data:

```
. use analysis
. xtset fips time
. spset fips
. spset, modify shpfile(shapefile_shp)
```

The above will work as long as *analysis*.dta does not contain counties that do not appear in *shapefile*_shp.dta; see shpfile() and drop under *Options* above.

Notice that spset expects xtset to handle panel-data details. With panel data, you are required to xtset the data first. After the spset, if you typed xtset without arguments, you would discover that the spset modified the xtset setting. Data that were xtset on fips and time will now be xtset on _ID and time. When you typed spset fips, spset created the variable _ID equal to fips, and then it changed the xtset setting to match its own. spset does not drop the variable fips; it just makes its own copy of it.

Actually, what we typed for the panel-data case may not be sufficient. We should have typed

Panel data:

```
. use analysis
. xtset fips time
. spbalance, balance                // <-- new
. spset fips
. spset, modify shpfile(shapefile_shp)
```

`spset` requires that panel data be strongly balanced. `spbalance, balance` will make panel data strongly balanced if they are not already. We omitted it because `spset` will verify that the data are strongly balanced and, if they are not, will issue an error. If `spset` complains, we can type `spbalance, balance` and then type the `spset` command again. See [SP] **spbalance**.

Whether data are cross-sectional or panel, you may need to modify the `coordsys()` setting. Use

All data:

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

It is important that the coordinate system be set correctly; see [SP] **spdistance**.

Setting data without a shapefile but with coordinates

Assume that *analysis*`.dta` is the same county dataset used previously. In addition to `fips` and perhaps `time`, the data also contain variables `x` and `y` recording the coordinates of each county.

To `spset` the data without a shapefile, type

Cross-sectional data:

```
. use analysis
. spset fips, coord(x y)
```

Panel data:

```
. use analysis
. xtset fips time
. spset fips, coord(x y)
```

If `x` contains longitude and `y` contains latitude, also type

All data:

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

Setting data without a shapefile

Assume that *analysis*`.dta` no longer contains variables `x` and `y`. We have no shapefile and no coordinates. At this point, the data are probably not even geographically based. So rather than `fips`, we will assume the spatial units are uniquely identified by `node`. If the data are panel data, we assume observations are identified by `node` and `time`.

To `spset` the data, type

Cross-sectional data:

```
. use analysis
. spset node
```

Panel data:

```
. use analysis
. xtset node time
. spset fips
```

Modifying settings

You use `spset, modify` to modify settings of data that are already `spset`. You may modify whether the data contain coordinates, whether the coordinates are planar or latitude and longitude, the ID-variable codes used to identify the spatial units, and whether the data are linked to a shapefile.

Modifying coordinates

The coordinates for each of the spatial units in your data are stored in variables `_CX` and `_CY` if Sp knows them.

Sp knows the coordinates if you are linked to a shapefile. It knows them because Sp itself calculated the centroids of the spatial units from the information in the shapefile and stored the results in `_CX` and `_CY`.

Sp also knows the coordinates if you are not linked to a shapefile but specified the coordinates when you originally `spset` the data. In that case, it copied the coordinates you supplied into `_CX` and `_CY`.

If you are linked to a shapefile, you may not modify the coordinates Sp has stored—nor would you want to modify them.

If you are not linked to a shapefile, you can add or replace coordinates by typing

```
spset, modify coord(xvar yvar)
```

If you want to delete the coordinates, type

```
spset, modify nocoord
```

Modifying how coordinates are interpreted

The coordinates stored in `_CX` and `_CY` are interpreted as planar or as degrees latitude and longitude. The interpretation determines how distances are calculated; see [SP] **spdistance**. You can change the interpretation by typing

```
spset, modify coordsys(planar)
spset, modify coordsys(latlong)
```

In the case of the `latlong` setting, you can specify the units to be used for distances, too:

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

When you set or reset `coordsys(latlong)`, kilometers are assumed.

Modifying the ID variable

Variable _ID identifies the spatial units in your data. Each unit has a different code. The codes could be 1, 2, and 3 or any set of numbers.

If you started with a standard-format shapefile, Sp used 1, 2, 3, and so on for _ID when you used spshape2dta to translate the file to Stata format. Perhaps you subsequently modified the coding stored in _ID. We did in [SP] **intro 4** when we showed you how to prepare data using a shapefile. We modified _ID to contain FIPS codes.

You can modify the codes stored in _ID at any time. The commands are as follows:

spset *newidvar*, modify	if you are not linked to a shapefile
spset *newidvar*, modify replace	if you are linked to a shapefile

Avoid doing this. These commands exist so that you can modify _ID at the outset when you are preparing your data. At that stage, you have no investment in the codes that are being used.

Later, you have an investment. The codes were used to identify the rows and columns of spatial weighting matrices you created. If you change the codes, any weighting matrices you have saved will become unusable.

If you are linked to a shapefile and change the codes, Sp will reindex both your data and its linked shapefile. If other datasets are linked to the same shapefile, their links to it will no longer work.

In [SP] **intro 4**, [SP] **intro 5**, and [SP] **intro 6**, we modified codes before you became invested in the coding system used.

Sometimes, you really do have to change codes later. Let's imagine the unimaginable situation where the U.S. Census Bureau changes from FIPS in favor of NEWFIPS. Even then, we would ask you whether you really need to migrate to NEWFIPS, but for this example we will assume that you do. We will assume that you have a migration dataset containing two variables, fips and newfips. Variable newfips is never missing, but some fips values might be. Start by taking the migration dataset and dropping any observations for which fips is missing:

```
. use migration, clear
. drop if missing(fips)
. save mymigration
```

Now, merge with your analysis file:

```
. use project, clear
. merge 1:1 fips using mymigration
. assert _merge!=1              // no master unmatched
. keep if _merge==3             // keep the matches
. drop _merge
```

You can now change Sp's _ID variable. If project.dta is not linked to a shapefile, type

```
. spset newfips, modify
. save, replace
```

If it is linked to a shapefile, type

```
. spset newfips, modify replace
. save, replace
```

File project.dta now uses NEWFIPS. There is no solution that will allow the use of old spatial weighting matrices indexed on FIPS. You will be using the NEWFIPS codes.

If your data were linked to a shapefile and you have other datasets linked to the shapefile you just reindexed, you need to do the following with each dataset:

```
. use dataset, clear
. spset, modify noshpfile
. merge 1:1 fips using mymigration
. assert _merge!=1              // no master unmatched
. keep if _merge==3             // keep the matches
. drop _merge
. spset newfips, modify
. spset, modify shpfile(shapefile_shp.dta)
. save, replace
```

Modifying whether the data are linked to a shapefile

The commands

```
spset, modify shpfile(shapefile)
spset, modify noshpfile
```

make and break links to shapefiles. When you establish a connection, variable _ID must use the same codes as the Stata-format shapefile *shapefile*.

We used these commands in the example in the previous section.

Converting cross-sectional data to panel data and vice versa

Cross-sectional data can become panel data and vice versa. A cross-sectional dataset could become panel because of a merge. A panel dataset could become cross-sectional because of a drop.

Here is a case of cross-sectional data becoming panel data:

```
. use analysis              // cross-sectional data
. spset
  Sp dataset
                  data:  cross sectional
     spatial-unit id:  _ID
         coordinates:  _CY, _CX (latitude and longitude, miles)
    linked shapefile:  shapefile_shp.dta
. merge 1:m fips using paneldata
  (output omitted)
```

Note that the data were `spset` before the `merge`, and after the `merge`, the data are panel data, but they are not yet `xtset`. If you typed `spset` without arguments right now, it would complain about repeated _ID values. To fix the problem, `xtset` the data:

```
. xtset fips time
```

Now, `spset` will report

```
. spset
  Sp dataset
                  data:  panel
     spatial-unit ID:  _ID
             time id:  time (see xtset)
         coordinates:  _CY, _CX (latitude-and-longitude, miles)
    linked shapefile:  shapefile_shp.dta
```

Now, let's convert these panel data back to cross-sectional data:

```
. keep if time==1
```

Here is how you tell Sp that the data are no longer panel data:

```
. xtset, clear
```

Now, spset will report

```
. spset
  Sp dataset
                 data:  cross sectional
    spatial-unit ID:  _ID
        coordinates:  _CY, _CX (latitude-and-longitude, miles)
   linked shapefile:  shapefile_shp.dta
```

Stored results

spset stores the following in r():

Macros

r(sp_ver)	1
r(sp_id)	_ID
r(sp_id_var)	*varname* or empty
r(sp_shp_dta_path)	path to _shp.dta file
r(sp_shp_dta)	*shapefile*_shp.dta
r(sp_cx)	_CX or empty
r(sp_cy)	_CY or empty
r(sp_coord_sys)	planar or latlong
r(sp_coord_sys_dunit)	kilometers or miles if r(sp_coord_sys) = latlong

Also see

[SP] **intro 3** — Preparing data for analysis

[SP] **intro 4** — Preparing data: Data with shapefiles

[SP] **intro 5** — Preparing data: Data containing locations (no shapefiles)

[SP] **intro 6** — Preparing data: Data without shapefiles or locations

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

[SP] **spbalance** — Make panel data strongly balanced

[SP] **spdistance** — Calculator for distance between places

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

[XT] **xtset** — Declare data to be panel data