

1 Organizing time series in Matlab structures

The first step is to store your time series and its metadata in structures with defined fields. This step assures uniformity in storage of the time series data of different students, and is necessary to ensure that all Matlab scripts and functions written for the course will run without the need for individually-tailored input instructions. A structure is a Matlab variable-type similar to a database in that the contents are accessed by textual field designators. A structure can store data of different forms. For example, one field might be a numeric time series matrix, another might be text describing the source of data, etc. In this first lesson you will run a Matlab script that stores your data in structures. In subsequent lessons you will apply time series methods to the data by running Matlab scripts and functions that operate on those structures. First you must organize your time series in Excel spreadsheets and put the metadata for the series in ascii files. This lesson describes the preparation of your time series for use in the course, and introduces computation and graphics with Matlab. These notes are supplemented by two files: [appendixa.pdf](#), which describes Matlab, the toolboxes needed, and how the software might be accessed; and [appendixb.pdf](#), which has instructions for formatting and submitting assignments.

1.1 Data types: V1, V2, V3

You must prepare three sets of time series: V1, V2 and V3. In bivariate analyses, series from V1 will be run against those from V2, and some of the analyses will assume a hypothetical “causal” system, in which series from V2 could physically “affect” those from V1. For example, precipitation might “affect” tree-ring indices, in a causal sense. If your data can be organized so, put the response variables in V1 and the variables bringing about the response in V2. In a systems view, V2 contains the “input” variables, and V1 the “output” variables. Series from V3 are intended for illustrating detrending, and do not need to be related to any of the series in V1 and V2. The data selected for V3 should be expected to have trend in mean. Examples of data choices in dendroclimatology are:

V1 = tree-ring chronologies
V2 = annual precipitation
V3 = measured ring widths for selected cores

Examples of choices in hydrology are:

V1 = gauged total flow for the water year
V2 = water-year total precipitation at stations
V3 = gauged total flow for rivers with anthropogenic influence increasing over time

Examples of choices in climatology are:

V1 = Palmer drought severity index
V2 = Temperature or precipitation
V3 = Temperature series affected by urban warming

The designation of “input” and “output” is “nominal” in that your time series might not fall neatly into a physical causal system. If not, arbitrarily group one set of time series as “input” series and another group as “output” series. The important thing to keep in mind is that series from V1 will be compared with those in V2 in bivariate analysis (e.g., correlation). Choose V1

and V2 so that at least one of the series in V1 is likely to be correlated with at least one of the series in V2.

Ideally the series in V1 and V2 are not dominated by trend. This is because some of the methods used on these series assume stationarity of mean.

1.2 Number of series in V1, V2, V3

Include at least three series in each of V1, V2 and V2. The maximum number of series in any of those structures is unimportant, but to avoid cumbersome screen menus make it no more than 12.

1.3 Time step and sample size

A time series is a variable observed at different times. A critical requirement for time series analysis in the course is a uniform time step, or time spacing of observations. The class examples and data sets used by most students have a time step of one year (tree-ring indices, summer-total rainfall, etc.). Other time steps are acceptable as long the time step is constant and the same for all time series. The time step should be the same for series in V1, V2 and V3.

Sample size is important for some time series applications. To avoid problems with inadequate sample size, all time series in V1 and V2 should have a common period (overlap) of at least 50 observations. The series in V3 are used independently of those in V1 and V2, and so are not required to overlap at all with V1 and V2 series. Moreover, it is not necessary that the individual series in V3 overlap one another.

1.4 Input format of time series

Put the time series for V1, V2, and V3 in three separate spreadsheet files, with suffix `xls`. Matlab function `geosal` will be used to read the data, and expects the data to be in the first sheet of each `xls` file. The first column of the spreadsheet should be the time variable (e.g., year) and the other columns should be the individual time series. The first row in the `xls` file should be an alphanumeric header line. For example, cell A1 might contain “Year”, and cell B1 might contain the name of the first series (series whose data is in column 2 of the spreadsheet). These headers from the spreadsheet are cross-checked against the metadata by `geosal` to make sure the order of time series is in the `xls` file matches the order of series listed in an associated metadata txt file (see below).

1.5 Missing data

The data selected for V1, V2, and V3 should be continuous in the sense that there are no internally missing values. If you want to use time series that have missing values, fill in those missing values by some appropriate method before storing the data in V1, V2 or V3.

The block of cells containing time series in the `xls` file can be viewed as a time series matrix. This matrix can have missing data (blanks, or some specified numeric code), but the missing data must be at the ends of the series, not internal. For example, the block of data might cover the years 1801-2006, one series might have valid data just for the portion 1830-1999, another for 1815-2001, etc. `Geosal` will interpret blanks in the spreadsheet as missing data. `Geosal` converts the spreadsheet data to a standard form for use in exercises in the course. In Matlab’s storage, the missing values are coded as “not-a-number”, or NaN. You do not have to use blanks as missing values in the spreadsheets. Any numeric code is acceptable (e.g., -9999), but be sure it could not possibly be confused with actual data.

1.6 File naming

A separate `xls` file must be prepared for each of the input data sets V1, V2, and V3 described in 1.1. For convenience, the filename should indicate whether the data is V1, V2 or V3. For example, you might have spreadsheets `v1arizona.xls`, `v2arizona.xls` and `v3arizona.xls`. The “arizona” part of the names in these examples would vary from student to student, and is an arbitrary name of your own choice for identifying the data. But stick with the convention of using “v1”, “v2” and “v3” as the first two characters of the filenames.

`Geosal` will also prompt for the name of the output `“_mat”` that will store your data for the course. The input data and metadata from V1, V2 and V3 will go into the `mat` file. The suffix `“.mat”` means the file is in Matlab’s native storage format. I recommend giving the `mat` file a name that associates the file with you. For example, if your name is Ramzi Touchan, and this is Spring 2011 semester, a logical name might be `“Touchan2011.mat”`.

1.7 Supporting information for time series—the metadata

Each of the three spreadsheet files (`xls` files) should be accompanied by an ASCII metadata file. This metadata is used for producing menu choices in various scripts and functions, and for labeling plots and text output. `Geosal` expects these ASCII text files, so you must build them beforehand. Any text editor that produces ASCII output (e.g., notepad) will do. The University of Arizona has a site license for TextPad, which is the text editor I currently use on my windows PC (<https://sitelicense.arizona.edu/nocost.shtml>).

The metadata `txt` files have a very simple structure, which you must follow exactly or else `geosal` will bomb. Each `txt` file consists of three initial lines followed by one line of information for each time series. The three initial lines apply to all series in the `xls` file, and are:

- 1 filename of the `xls` file with the time series
- 2 time step for the data
- 3 missing value code used in `xls` file

Subsequent lines, one per time series, give the following information, with elements on a line separated by a dollar sign, `“$”`:

- 1 short name, for labeling series, 15 or fewer characters
- 2 long name of series, 40 or fewer chars
- 3 y-axis label for variable (type of data), 20 characters or fewer
- 4 units of variable, 13 or fewer characters

The short name should be sufficient for you to identify series in plot labels, and should not contain any internal blanks or underscore (`“_”`) characters. The long name can have more descriptive information on the series, and can contain internal spaces. The y-axis labels and the “units” descriptor should not contain internal spaces or underscores.

1.8 Sample input files

A simple example using fictional data will serve to illustrate the format and content of the input data for script `geosal.m`. Consider a V1 data set consisting of 6 tree-ring index chronologies. Assume you have already stored these time series in a file “`v1spring11.xls`”. Your `txt` metadata file, `v1spring11.txt`, might look like this:

```
v1spring11.xls
Year

MEAF-PSME $ Mesa Alta Fir PSME, standard index      $ Index $ Dimensionless
MEAP-PIST $ Mesa Alta Pine PIST, standard index     $ Index $ Dimensionless
BCWF-PSME $ Bear Can W Fir PSME, standard index    $ Index $ Dimensionless
BCWP-PIST $ Bear Can W Pine PIST, standard index   $ Index $ Dimensionless
FEN-PIPO  $ Fenton Lake PIPO, standard index       $ Index $ Dimensionless
EAU-PSME  $ Echo Amphitheater PSME, standard index $ Index $ Dimensionless
```

The first line tells `geosal.m` the name of the file with the spreadsheet of V1 series. The second line says the time step for all time series is a “Year”. The third line is blank, meaning that a blank cell is interpreted as missing data in the spreadsheet. Remaining lines list the individual V1 time series. For example, the first series has id “MEAF-PSME”, long name “Mesa Alta Fir PSME, standard index”, and is an “Index” that is dimensionless – has units “Dimensionless”.

For this example, files `v1spring11.xls` and `v1spring11.txt` apply to the V1 data. You also need to prepare corresponding pairs of `xls` and `txt` files for the V2 and V3 data.

1.9 Running `geosa1.m`

`Geosal` is a function that reads the time series and metadata (the `xls` files and `txt` files) and stores the data in a `mat` file that can be used in the rest of the course. `Geosal` stores all the data in a single `mat` file. Before running `geosal`, make sure that the required input data files are in the current Matlab working directory (see `appendixa.pdf`). Detailed instructions for running `geosal` are found elsewhere (see “**Running `geosa1.m`**” in `a1.pdf`).

1.10 Checking that the structure variable has been created and stored

After running `geosal`, you should have a `mat` file (e.g., `spring09.mat`) in your current working directory, and that `mat` file should hold `vlist` (a list of variables in the `mat` file), and a subset of time series structure variables V1, V2, V3. Say the `mat` file is named `spring11.mat`. Check out contents of the `mat` file by typing commands:

```
>> what
```

Shows a list of `mat` files in the command window. You should see `spring11.mat` in the list

```
>>load spring09.mat
```

Clears the workspace and load the .mat file you have created. Of course, you will have given your file a different name than `spring11.mat` (e.g, `Pushkin2011.mat`)

```
>> whos
```

lists variables in the workspace. You should see `vlist` and one or more of `V1`, `V2`, `V3` there now.

```
>>vlist
```

lists the contents of variable `vlist`

```
>>V1
```

Typing a variable's name displays the contents of the variable in the command window. Names are CASE SENSITIVE. When you type `V1`, you see a list of all the fields in the structure variable `V1`.

```
>>V1.id
```

Type the structure followed by a period and a fieldname and you get the expanded contents of the field. So, for example, lists the short series names, or `ids`

```
>>V1.name
```

lists the names long series names

```
>>V1.tsm
```

lists the time series themselves.

Note that `V1.tsm` has only the time series data, not the vector of years. `V1.time` is the year vector. You can generate a quick time series plot of the first and third series, with a legend of `ids`, by entering this sequence of commands

```
>>t=V1.time;
>>x1=V1.tsm(:,1);
>>x3=V1.tsm(:,3);
>>plot(t,x1,'-',t,x3,'-');
>>ylabel(V1.label{1});
>>xlabel(V1.increment);
>>legend(V1.id{1},V1.id{3});
```

The first three commands rename variables so that the plot syntax is less cumbersome. The next commands plot the series and label the plot. The next command generates the plot. The last three commands label the plot and add a legend. Entering the commands one by one at the command prompt is one way to use Matlab. A better way is to organize the commands in an ascii file called a "script" and then run the script from the command prompt. Using scripts avoids having to re-enter commands each time you want to repeat an analysis or modify some part of the analysis or the plots. The interactive tools at the top of the figure window are another way to change attributes of plots. Please refer to the Matlab help files for more information on graphics.

Matlab functions, like scripts, are also ascii files containing commands, but are more general in that they may be applied to different problems by changing the input arguments. The `plot` command above is a function with input arguments the x-data, y data and line-type for the plot. Matlab has many built-in functions for data analysis, and you can extend this capability with your own functions or user-written functions available over the Web.

It is helpful to learn some basic elements of Matlab programming if you want to apply Matlab functions other than those covered in the course, or if you want to tailor your own data analysis. But you do not need to program in Matlab for this course. The script files and functions needed for all analyses have already been written and are provided for the course.

1.11 Turning in assignments as figures and captions

Assignments for the course consist of running Matlab scripts or functions demonstrating the time series methods and writing up brief interpretations of the results. Each assignment should be turned in as a pdf document with figures and captions only. The figures are generated by the Matlab scripts. The captions are your answers to the questions in the assignment. Please refer to file `appendixb.pdf` (“Appendix B -- Submitting Assignments”) for instructions on preparing and submitting assignments.