X-13 Toolbox for Matlab

By Yvan Lengwiler
University of Basel, Switzerland
yvan.lengwiler@unibas.ch

October 03, 2017, Version 1.33

Please reference the software as

> Yvan Lengwiler, 'X-13 Toolbox for Matlab, Version 1.33', Mathworks File Exchange, 2017, http://ch.mathworks.com/matlabcentral/fileexchange/49120-x-13-toolbox-for-seasonal-filtering

if you use it for your reports or publications. It's only fair to give proper credit :)

NOTE: This version of the toolbox works with version 1.1 build 39 of the X-13 programs of the Census bureau. If you have older versions of the Census programs, be sure to issue the **InstallMissingCensusProgram** command before using this toolbox.

Also, if you have an older version of the X13 Toolbox for Matbab, it is recommended that you delete it first (or rename the directory containing the old toolbox). Some of the files have changed location in the new version and having an old and a new version of the same file in different branches of your path can lead to problems.

Contents

InstallMissingCensusProgram

```
InstallMissingCensusProgram installs pieces of software from the US
Census Bureau that are necessary to perform the seasonal filtering.

Usage:
  InstallMissingCensusProgram()
  InstallMissingCensusProgram(arg, [arg2], [...])
  success = InstallMissingCensusProgram([...])
  InstallMissingCensusProgram('all')

If called with no argument, the program tries to install all usable
files. Alternatively, an argument or a list of arguments can be provided.
Choices are:
  'x13prog'       X-13 software, ascii and html versions
  'x13doc'        documentation of X-13 program
  'x12diag'       X-12 diagnostic utility
  'x12prog'       X-12 software, 64 bit and 32 bit versions
  'x12doc'        documentation of X-12 program
The function returns a vector of booleans, indicating which installations
were successful.

Using this function with no arguments
  InstallMissingCensusProgram;
should produce the following result:
  Downloading 'x13as_V1.1_B39.zip' from US Census Bureau website ... success.
  Downloading 'x13ashtml_V1.1_B39.zip' from US Census Bureau website ... success.
  Downloading 'docsX13.zip' from US Census Bureau website ... success.
  Downloading 'gettingstartedx13_winx13.pdf' from US Census Bureau website ...
  success.
  Downloading 'docsX13Acc.zip' from US Census Bureau website ... success.
  Downloading 'itoolsv03.zip' from US Census Bureau website ... success.
  Downloading 'omega64v03.zip' from US Census Bureau website ... success.
  Downloading 'omegav03.zip' from US Census Bureau website ... success.
  Downloading 'docsv03.zip' from US Census Bureau website ... success.
  Downloading 'gettingstartedx12.pdf' from US Census Bureau website ... success.

After that, all programs of the US Census Bureau website that are
supported by the X-13 Toolbox are installed on your computer.

In addition, five other programs that are related to X13ARIMA-SEATS can also
be downloaded:

  'x11prog'       An early version of the Census program.
  'x11doc'        Some documentation of the easrly X-11 version.
  'genhol'        A program that allows the user to create variable files for
                  holidays.
  'x13graph'      The JAVA version of the X-13 graph program.
  'x13data'       A utility to transform data in an Excel sheet into files
```

	usable by x13as.exe, as well as for collecting x13as.exe output and storing it in an Excel file.
'cnv'	A utility to convert X-12 specification files to the X-13 format.
'sam'	A utility to change several spec files at once. It's unlikely that you will use this if you use X-13 with Matlab.

These programs are not directly supported by the Matlab-Toolbox. x11 is not supported because its syntax is completely different from x13 and it is potentially prone to Y2K problems. The x13graph java program can be used if you add the 'graphicsmode' switch when calling x13, but you need to start the graph program from the command line. Likewise, genhol, or the version with a GUI called wingenhol, can be used independently from Matlab. The toolbox does not interact with these programs directly.

Calling InstallMissingPrograms('all') installs everything, including these six additional sets of programs and files.

guix

guix is a GUI that allows you to easily create a specification for a seasonal adjustment for the X-13ARIMA-SEATS program (or the X-12-ARIMA program), perform the computations, and view the results in the GUI. guix supports only single time series (x13series), not composites (x13composite).

Usage:

```
guix
guix('style')
guix 'style'
guix('variable')
guix variable
h = guix([style],['variable'])
```

Inputs and Outputs:

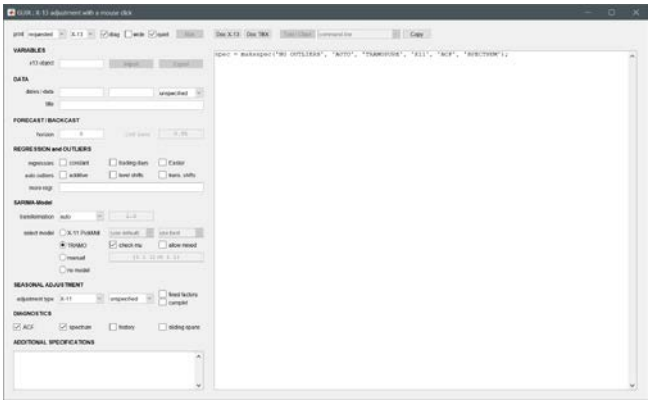
'style'	Can be 'normal', 'modal', or 'docked' (or 'n', 'm', or 'd', for short), indicating the WindowStyle of the GUI. Default is 'normal'.
variable	A x13series variable in the main workspace.
'variable'	The name of a x13series variable in the main workspace, as string.
h	A struct containing handles to the GUI and to its components.

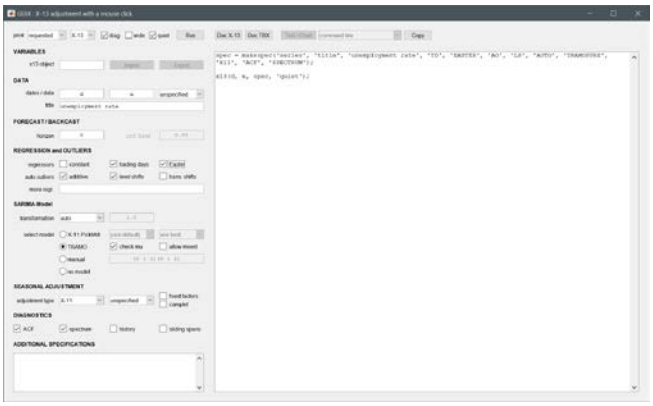
Usage of this GUI should be self-explanatory (if you are familiar with the outputs that X-13ARIMA-SEATS generates). In the dates / data field, the user can enter variable names containing the dates (a vector) and the data (also a vector) that are to be worked on (also a vector). These variables must be present in the calling workspace. You can also import an x13series object existing in the calling workspace into guix with the 'Import' button.

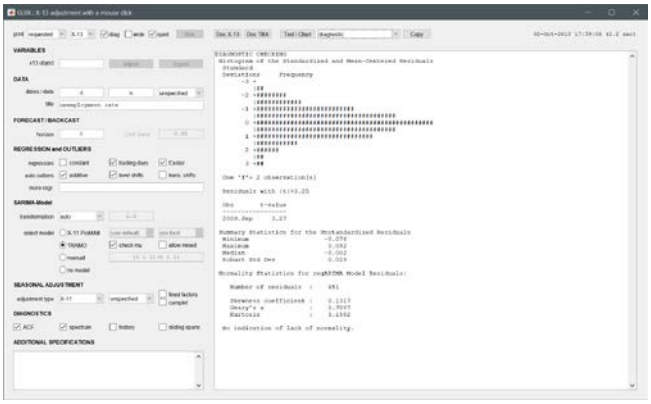
The 'Run' button performs the computations. You can export the resulting x13series object to the calling workspace with the 'Export' button.

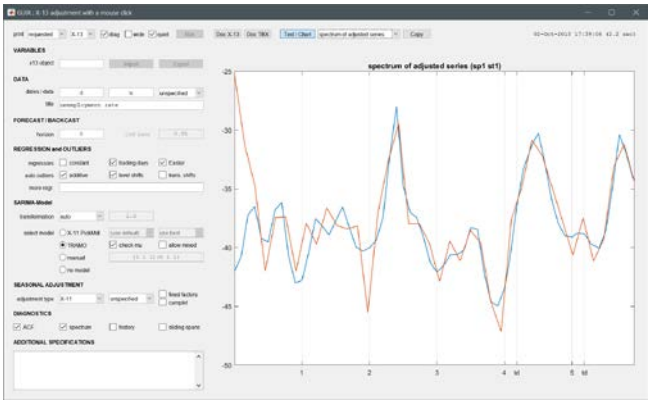
With the 'Text/Chart' button you can switch between viewing tables and text items on the one hand, and plots on the other.

The 'Copy' button copies the current output window (text or chart) to Windows' clipboard. You can then paste it into some other program.








spectrum of adjusted series (sp1 sit1)

x13

```
x13 calls the x13as program of the US Census Bureau / Bank of Spain to perform
seasonal and extreme value adjustments.

Usage (single time series):
  x = x13([dates,data])
  x = x13([dates,data],spec)
  x = x13(dates,data)
  x = x13(dates,data,spec)
  x = x13(ts,spec)
  x = x13(..., 'quiet')
  x = x13(..., 'x-12')
  x = x13(..., 'x-13')
  x = x13(..., 'html')
  x = x13(..., '-s', '-c', '-w', ... etc)
  x = x13(..., 'noflags')
  x = x13(..., 'graphicsmode')
  x = x13(..., 'graphicsloc',path)
  x = x13(..., 'fileloc',path)
  x = x13(..., 'progloc',path)
  x = x13(..., 'prog',filename)

'dates' and 'data' are single column or single row vectors with obvious
meanings. 'dates' must contain dates as datenum codes. Alternatively, use
a timeseries object ('ts') containing a single time series. In version 3 and 4
above, dates can also be a datetime class variable (this is available in ML
2014b and later).

'spec' is a x13spec object containing the specifications used by the
x13as.exe program. If no 'spec' is given, the program uses the specification
that is produced by makespec('DEFAULT'), see help makespec.

The output 'x' is a x13series object containing the requested numerical
results, as well as the data and dates used as input.

Normally, all warnings of the x13as/x12a program are shown on the console as
Matlab warnings (for instance when a variable was requested but is not
available). The switch 'quiet' suppresses warnings. The corresponding
messages will still be contained in the resulting object, but they will not
show up on the screen at runtime.

Three switches are available that determine the program that is used to
perform the seasonal decomposition. The 'x-12' switch uses the x12a program
instead of the x13as program. The 'x-13' switch enforces the use of the x13as
program. This is the default. Note that the x13spec object allows all settings
known to x13as, but this is a superset of what is known to x12a. Therefore,
one can easily create settings that are not compatible with x12a. In
particular, makespec will often create specifications that do not run with
```

x12a. (As an example, the keys 'spectrum' and 'seats' are not known to x12a.) Extra care is therefore required when using x12a.

The 'html' switch uses the 'accessible version' of the Census program. The accessible version formats the tables and log files in html. Using this version has the advantage that you can view the output neatly formatted in your browser. The disadvantage is that the tables are not extracted and placed into the x13series (or x13collection) object. So, x.listoftables and x.table are empty. Instead, you can inspect the tables in the browser with web(x.out) and web(x.log). Note that 'html' has no effect if the 'x-12' or 'prog' options are used.

Any string arguments starting with a hyphen are flags passed on to x13as.exe through the command line. Section 2.7 of the X-13ARIMA-SEATS Reference Manual explains the meanings of the different flags. Some flags are dealt with by the x13 Matlab program, so they should not be used by the user (in particular, using -g, -d, -i, -m, or -o is likely to mess up the functioning of the program).

The most relevant flags are
-s Store seasonal adjustment and regARIMA model diagnostics in a file
-t Same as -s
-n (No tables) Only tables specifically requested in the input specification file will be printed out
-r Produce reduced X-13ARIMA-SEATS output (as in GiveWin version of X-13ARIMA-SEATS)
-w Wide (132 character) format is used in main output file
-c Sum each of the components of a composite adjustment, but only perform modeling or seasonal adjustment on the total
-v Only check input specification file(s) for errors; no other processing
-q Run X-13ARIMA-SEATS in quiet-mode (warning messages are not sent to the console). This is equivalent to the 'quiet' switch.

The -q flag as defined by x13as.exe suppresses all messages. This is redefined in this toolbox to just suppress the showing of these messages on the console. They are still generates and available in the x12series object. In fact, setting the -q flag is equivalent to using the 'quiet' switch.

To use flags, use one of the following syntaxes,
```
   x = x13(..., '-s'), or
   x = x13(..., '-s', '-w'), or
   x = x13(..., '-s -w'),
or x = x13(..., 'noflags')
```
If no flag is set by the user, the default is to set the '-s -n' flags, so that the diagnostics file is generated (-s) and only the requested tables (-n) are written to the .out property. The 'noflags' option removes all the flags, including the default.

Four optional arguments can be provided in name-value style: The argument

following the 'fileloc' keyword is the location where all the files should be stored that are used and generated by the x13as program. If this optional argument is not used, these files will be located in a subdirectory 'x13' of the system's temporary files directory (%tempdir%\x13).

The argument following the 'graphicsloc' keyword is the location where all the files should be stored that can be used with the separately available X-13-Graph program (see https://www.census.gov/srd/www/x13graph/). If this optional argument is used, x13as will run 'in graphics mode' and these files will be generated. If this argument is not used or set to [], the graphics-related files will not be generated. If 'graphicsloc' is set to '' (i.e. an empty string), then the graphics files will be created in a subdirectory called 'graphics' of the fileloc directory. The same is achieved with the switch 'graphicsmode'.

The arguments following 'progloc' and 'prog', respectively, allow you to specify the location of the executables that do the computations. 'progloc' indicates the folder where the executables can be found. By default, this is the 'exe' subdirectity of the X-13 toolbox. 'prog' is the name of the executable itself. By default, this is 'x13as', or 'x12a' (or 'x12a64' on a 64-bit computer) if the 'x-12' option is set.

So what is the 'prog' option really used for? You could, in principle, specify alternative executables, other than the ones provided by the US Census Bureau. The output of such an alternative program would have to be compatible with the output generated by the Census Bureau program. So, in practice, the only two conceivable options here are that either you have an older version of the Census Bureau software that you want to use, or you have a beta version which is in development. For instance, a previous version of x13as was version 1.1 build 9. If you have a copy of it, and you called it 'x13asv11b9.exe' on your harddisk (in the exe subdirectory of the toolbox), then

```
  x = x13(..., 'prog','x13asv11b9');
```

uses the previous version of the Census program.

Usage (composite time series):

```
  x = x13([dates1,data1],spec1, [dates2,data2],spec2, [...], compositespec)
  x = x13( dates1,data1 ,spec1, dates2,data2  ,spec2, [...], compositespec)
  x = x13([dates,data]  ,{spec1,spec2,...},compositespec)
  x = x13( dates,data   ,{spec1,spec2,...},compositespec)
  x = x13(ts,{spec1,spec2,...},compositespec)
  x = x13(ts,spec1,[dates,data],spec2,dates,data,spec3,compositespec)
  x = x13(..., 'x-12')
  x = x13(..., 'x-13')
  x = x13(..., 'html')
  x = x13(..., '-s', '-c', '-w', ... etc)
  x = x13(..., 'noflags')
  x = x13(..., 'graphicsmode')
  x = x13(..., 'graphicsloc',path)
  x = x13(..., 'fileloc',path)
```

```
  x = x13(..., 'progloc',path)
  x = x13(..., 'prog',filename)
```

In the first and second version you can set different specifications for the individual time series. Alternatively, in the third and fourth usage form, 'data' may be an array with m columns, where each column is interpreted as one timeseries. In the fifth usage form, all variables in the timeseries object 'ts' are interpreted as time series of the composite run. You can also combine the syntax as seen in the sixth usage form.

For composite runs, the last argument (except possible optional arguments) is always the specification of the composite series ('compositespec' cannot contain the 'series' section, but must contain the 'composite' section).

Example 1:

```
  spec = makespec('DIAG','PICK','X11');
  x = x13([dates,data],spec);
```

Then, 'x' is a x13series object with several variables, such as x.dat (containing the original data), x.d10, x.d11, x.d12, x.d13 (containing the results of the X11 filtering as produced by the x13as program), as well as different tables (essentially plain text). See the help of x13series for further explanation.

Example 2:
Let C, I, G, NX now be components of components of GDP of a country, measured at quarterly frequency (Y=C+I+G+NX), and D be the common dates vector when the components were measured.

```
  spec = makespec('AUTO','TRAMO','SEATS','series','comptype','add');
  x = x13( ...
    [D,C], x13spec(spec,'series','name','C'), ...
    [D,I], x13spec(spec,'series','name','I'), ...
    [D,G], x13spec(spec,'series','name','G'), ...
    [D,NX],x13spec(spec,makespec('ADDITIVE'),'series','name','NX'), ...
    makespec('AUTO','TRAMO','SEATS','composite','name','Y'));
```

Then, 'x' is a x13composite object containing x13series C, I, G, NX, and Y. Again, see the help of x13composite and x13series for further explanation.

Alternatively, you can produce the same result as follows:

```
  spec = makespec('AUTO','TRAMO','SEATS','series','comptype','add');
  allspec = { ...
      x13spec(spec,'series','name','C'), ...
      x13spec(spec,'series','name','I'), ...
      x13spec(spec,'series','name','G'), ...
      makespec(spec,'ADDITIVE','series','name','NX')};
  compspec = makespec('AUTO','TRAMO','SEATS','composite','name','Y');
  x = x13(D,[C,I,G,NX],allspec,compspec);
```

Requirements: The program requires the X-13 programs of the US Census Bureau to be in the directory of the toolbox. The toolbox attempts to download the

required programs itself. Should that attempt fail, you can download this software yourself for free from the US Census Bureau website. Download http://www.census.gov/ts/x13as/pc/x13as_V1.1_B26.zip and unpack the x13as.exe file to the 'exe' subdirectory of this toolbox.

To install all programs and tools from the US Census Bureau that are supported by the X-13 Toolbox, issue the command InstallMissingCensusProgram once. The program will then attempt to download and install all files in one go.

Acknowledgments: Detailed comments by Carlos Galindo helped me make the Toolbox backward compatible.

x13series

x13series is the class definition for x13series objects.
Such an object is the home of the input to and the output of the US Bureau of the Census X13ARIMA-SEATS program as applied to a single time series.

Properties:

- name	string	name of the series
- filename	string	name of the files associated with the series
- fileloc	string	path to location of data files
- graphicsloc	string	path to location of files for the x13graph program; if this is an empty string (''), the graphics files are created in a subdirectory of the temporary files directory; if this property is empty ([]), no graphics files are produced by the .Run method.
- flags	string	flags to be used in the x13as run. Do not set the -g or the -m flags here; they are taken care of automatically. You could, for instance, set the -r or the -n flags to affect the .out property, and set the -s switch to generate the diagnostics summary file.
- specgiven	x13spec	specification provided when calling the x13 function
- spec	x13spec	specgiven is adapted by the program to remove inconsistencies. Also, some entries are added automatically. obj.spec the the final specification that is used for the estimation by x13as.
- isLog	boolean	True is decomposition is multiplicative, false if additive, empty if something else or unclear.
- period	int	periodicity (typically 4 or 12)
- span	string	dates spanned by variable
- arima	string	specification of seasonal ARIMA model
- coef	struct	stuct with one to three elements. obj.coef.arma contains the coefficients, standard errors, and t-values of the estimated ARMA process, obj.coef.regr contains the

```
                                    same for the preadjustment
                                    regression (if present). The two
                                    elements are Matlab tables and are
                                    available only with R2013a or
                                    later. obj.coef.lks contains the quality
                                    statistics (likelihood and the like) of
                                    the regression in a struct for easy
                                    access.
- regression       string           result of regression as text
  (removed, use .table('regression')  (extracted from .out)
- prog             string           name of executable used for the
                                    computation
- progloc          string           path to the x13as/x12a program
- ishtml           boolean          false if text version of executable
                                    is used, true if html version is
                                    used (in that case, obj.tbl is
                                    empty)
- progversion      string           version and build number of the
                                    Census program used
- timeofrun        1x2 array        time of running of program,
                                    duration of run
- con              string           console output of x13as.exe
- msg              string           errors, warnings, and notes during
                                    run
- listofitems      array            names of variables, ACF/PACF,
                                    spectra, and text items in the
                                    object (the items themselves are
                                    not hard-wired, but are dynamic
                                    properties)
- requesteditems   array            list of items that are requested in
                                    the specification file (with some
                                    'save' key)
- hitem            cells            array of handles to the dynamic
                                    properties
- tbl              struct           content of tables stored in the
                                    .out property
- version          double           version of the toolbox

.spec, .prog, .progloc, .ishtml, .fileloc, .graphicsloc, .flags, and
.timeofrun are freely accessible properties (they can be read and set
from anywhere). The other properties are either protected or dependent,
which means that you cannot easily set them (e.g., setting
obj.period = 12 throws an error).

Important methods:
- disp and display    Show the content of the object (extensive and
                      compact versions).
- dispstring          Same as disp, but does not print to the console.
                      Instead, the disp output is returned as a string
```

variable.
- plot — An overloaded method for this object class.
- table — Returns a particular table. table(obj,'F2A') returns table F2.A. table(obj,'F2') returns all tables starting with 'F2' (i.e. F2 and F2.A to F2.I) as one string. If no argument is given, a list of all tables in the object is returned.
- showmsg — Returns the content of the .msg property (which is a cell array) as a string.

Rarely used methods: The following methods are normally not useful for regular users. They are used by x13.m to perform its work. Be careful if you employ these methods. It is possible to create unusable x13series objects if you don't know what you are doing.

- additem — Used to add a general item to the object, obj = obj.additem(name,content), where name is a string with at most three letters.
- addvariable — Adds a time series to the object. Note, however, that time series must have names with at most three characters. Syntax: obj = obj.addvariable(vname, ... dates,data,header,type), where vname is the name of the new item (at most three characters), dates is a (nx1) column vector containing datecodes, data is an (nxm) array containing the data, header is a (1xm) cell array containing the titles of the individual series (if there is only one variable, m = 1, it is a good idea to use the name of the variable as the single header), and type is an integer with this meaning: type = 1 is a time series, type = 2 is an ACF/PACF object, type = 3 is a spectrum. type = 0 would be a text objects but you should use additem to add such contents.
- rmitem — Removes an item or a list of items (or variables) from the object.
- addtable — Takes two arguments: the name of the table and the content. Creates new table and places it in the object.
- rmtable — Removes the given table.
- PrepareFiles — Takes four arguments: dates, data, spec, and a boolean called isComposite which determines if the series is supposed to contain the composite series of a composite run (the members of a composite have isComposite set to false). It adds the items .dat and .spc to the object and prepares all the files on disk so that the x13as program can process them.
- Run — Runs the x13as program using the files created by PrepareFiles.
- CollectFiles — Imports the files produced by the x13as program into the Matlab object.
- RunFixedSeas — Runs the fixedseas function and packs the result

	into an x13series object.
- RunCamplet	Runs the camplet function and packs the result into an x13series object.
- RunX11	Runs a simplidied implementation of the X11 algorithm and packs the result into an x13series object.
- clean	Removes all the information that was added by CollectFiles.
- runX12diag	Runs the X-12 diagnostic utility on the files created with the -s flag.
- updatemsg	Extracts all ERRORS, WARNINGS and NOTES from the .err property and places them in the .msg property. Also adds a list of variables that were requested in the specification (with some 'save' key) but that are not available (because the x13 program did not produce them, or because they were later deleted).
- updatetables	Attempts to parse .out and puts the result into .tbl
- ExtractSection	Returns the content of a section in the .spc property.
- ExtractValue	Returns the value of a certain key in a certain section of the .spc property.

x13composite

x13composite is the class definition for x13composite objects.
Such an object is the home of the input to and the output of the US Bureau of the Census X13ARIMA-SEATS program as applied to a composite time series.

Properties:

- name	string	name of the series
- filename	string	name of the files associated with the series
- fileloc	string	path to location of data files
- graphicsloc	string	path to location of files for the x13graph program; if this is an empty string (''), the graphics files are created in a subdirectory of the temporary files directory; if this property is empty ([]), no graphics files are produced by the .Run method.
- flags	string	flags to be used in the x13as run. Do not set the -g or the -m flags here; they are taken care of automatically. You could, for instance, set the -r or the -n flags to affect the .out property, or set the -s switch to generate the diagnostics summary file.
- spec	x13spec	specification structure for estimation
- period	int	periodicity (4 or 12)
- span	string	dates spanned by variable
- prog	string	name of executable used for the computation
- progloc	string	path to the x13as/x12a program
- ishtml	boolean	false if text version of executable is used, true if html version is used (in that case, obj.table is empty)
- progversion	string	version and build number of the Census program used
- timeofrun	1x2 array	time of running of program, duration of run
- con	string	console output of x13as.exe
- msg	string	errors, warnings, and notes during run
- listofseries	array	names of x13series objects stored in this object

- compositeseries	string	name of x13series in the object containing the composite
- alldates	array	union of all .dat dates vectors
- hseries	array	handles to series in object

.spec, .prog, .progloc, .fileloc, .graphicsloc, and .timeofrun are freely accessible properties (they can be read and set from anywhere). The other properties are either protected or dependent, which means that you cannot easily set them (e.g., setting x.period = 12 throws an error).

In addition, x13series objects are added as new properties during an x13 run. These new properties contain the runs of the individual series that make up the composite, as well as the aggregated time series.

Important methods:

- disp and display	Show the content of the object.
- dispstring	Same as disp, but does not print to the console. Instead, the disp output is returned as a string variable.
- plot	An overloaded method for this object class.
- showmsg	Returns the content of the .msg property (which is a cell array) as a string.

Rarely used methods: The following methods are normally not useful for regular users. They are used by x13.m to perform its work. Be careful if you employ these methods. It is possible to create unusable x13series objects if you don't know what you are doing.

- PrepareFiles	Takes four arguments: dates, data, spec, and compSpec. data is the vecor for dates, data is the collection of series (the components), spec is the collection of x13spec specifications for the components, and compSpec is the specification for the composite series. The method calls the PrepareFiles method for the individual x13series objects.
- Run	Runs the x13 program using the files created by PrepareFiles.
- CollectFiles	Imports the files produced by the x13 program into the Matlab object.
- runX12diag	Runs the X-12 diagnostic utility on the files created with the -s flag.
- updatemsg	Extracts all ERRORS, WARNINGS and NOTES from the .err property and places them in the .msg property. Also adds a list of variables that were requested in the specification (with some 'save' key) but that are not available (because the x13 program did not produce them, or because they were later deleted).

x13spec

```
x13spec is the class definition for x13spec objects. Such an object is
used to set all specifications of a run of the X13-ARIMA-SEATS program.

Usage:
  Specifications are entered as triples: section-key-value.
  spec  = x13spec(section,key,value, section,key,value, ...);
  spec2 = x13spec(spec1, section,key,value, section,key,value, ...);
  spec3 = x13spec(spec1, section,key,value, spec2, section,key,value, ...);

Remark 1: section-key-value syntax ------------------------------------
spec = x13spec('series','name','rainfall','transform','function','auto')
would set name = rainfall in the series section, and function = auto in the
transform section. When using this with x13.m, this creates the following
.spc file on the harddrive:
  series{
      name = rainfall
  }
  transform{
      function = auto
  }
which is then used by the x13as.exe program.

Remark 2: merging existing specs --------------------------------------
If existing x13spec objects are entered as arguments (second and third
usage form above), the specifications are merged, from left to right,
i.e. later section-key-value pairs or settings in later specs overwrite
earlier ones.
Example:
  spec1 = x13spec('series','name','rainfall','x11','save','d10');
  spec2 = x13spec(spec1,'series','name','snowfall');
then spec2 contains save = d10 in the x11-section (inherited from spec1),
but name = snowfall in the series-section (the name rainfall was
overwritten).

Remark 3: accumulating keys -------------------------------------------
The keys 'save', 'savelog', 'print', 'variables', 'aictest', 'types',
'user', and 'usertype' behave differently. These keys are accumulated,
  spec = x13('x11','save','d10');
  spec = x13(spec,'x11','save','d11');
This does not overwrite the 'd10' value. Instead, 'd11' is added to the
list of variables that ought to be saved, and spec contains
save = (d10 d11) in the x11-section. To remove an item from one of these
special keys, use the RemoveRequests function. There are also
ExtractRequests, AddRequests, and SaveRequests methods.

Remark 4: entering empty sections -------------------------------------
An empty section can be added by specifying an empty cell for the key,
```

```
e.g. spec = x13spec('x11',{},{}) produces the entry
  x11{ }
in the .spc file.

Remark 5: removing sections or keys from a spec ------------------------
To remove a section completely, use an empty set in place of the key,
i.e., if spec has a 'x11' section, then spec = x13spec(spec,'x11',[],[])
removes the x11 section completely from this spec.

To remove a key from a section, use an empty set as value, as follows:
spec = x13('x11','save','d10','x11','savelog','q') produces
  x11{
      save = d10
      savelog = q
  }
Then spec = x13(spec,'x11','save',[]) removes the 'save' key and produces
  x11{
      savelog = q
  }
spec = x13(spec,'x11','save',{}), on the other hand, leaves the value of
x11-save unchanged.

Remark 6: user-defined variable ----------------------------------------
The 'regression' and 'x11regression' sections allow the user to specify
exogenous variables in the regressions that are not built in (like Easter
or TD or AO2003.Jan). The names of such variables are added with the
'user' key, the type of the variables is specified with the 'usertype'
key, and the exogenous variables themeselves are provided either with the
'data' key (in which case the data are part of the spec), or they are
defined in an extra file and then the name of the file is specified with
the 'file' key. You can use 'user', 'usertype', and 'data' in this
fashion with x13spec. You could also use the 'file' key, but in that case
you would have to make sure that your variables are stored as a table in
plain ascii text in a file and then provide the path to this file in the
spec. All of this is rather cumbersome.

For this reason, x13spec provides a more convenient way. Suppose your
exogenous variable is called 'strike' and is in your Matlab workspace.
You can then simply say
spec = x13spec(..., 'regression','user','strike', ...);
The program will then create a file filename.udv containing the strike
data in a form that is readable by the x13as program, and also adds the
correct entries to the spec-file.

If you have more than one user-definied exogenous variable, use this
syntax,
spec = x13spec(..., 'regression','user','(strike oilprice)', ...);

Remark 7: error checking -----------------------------------------------
```

x13spec allows you to set only sections that are known to the x13as program, and keys fitting to the respective sections. It does not check, however, if the values you assign are legal. If you assign illegal values you are likely to throw a runtime error by x13as.exe.

For an explanation of all available options and settings, consult the documentation of the x13as program provided by the US Census Bureau.

There is also a method
```
   spec.enforceX12spec
```
that removes sections or keys that are valid in X-13 but unknown to X-12. This is useful if you want to use the 'x-12' option in x13.m, and is in fact automatically applied internally when the 'x-12' option is set.

There also exist a few keys in the series and collection sections that are known to X-12, but not supported by X-13. For instance, X-12 supports
```
   series{spectrumstart = date value}
```
but X-13 does not support the key 'spectrumstart' in the 'series' section. For this reason, this toolbox does NOT SUPPORT these X-12 only section-keys.

Remark 8: short vs long names of saveable variables ----------
CAUTION: USE ONLY THE THREE-LETTER CODES FOR THE 'SAVE' KEY.
The x13as program uses a long name and a short three-letter name for variables or tables (e.g. 'save = levelshift' in the .spc file is equivalent to 'save = ls'). For the 'save' key, the Matlab X-13 toolbox recognizes ONLY the short two-or-three-letter versions of these variable names,
```
  x13spec('regression','save','ls')
```
Using the long name,
```
  x13spec('regression','save','levelshift')
```
may cause problems and is not advised.

Remark 9: pickmdl file lists ----------
If the X-11 'pickmdl' method is used to select the regARIMA model, a list of models to choose from should be supplied. You can create such a model list file yourself, or use one of the files provided for you by the toolbox. The selection of these ready-to-use model files includes:

- StatisticsCanada.pml — The default of Statistics Canada, contains 5 models.
- Hussain-McLaren-Stuttard.pml — 5 models proposed by these authors.
- ONS.pml — Default of the Office of National Statistics, United Kingdom. 8 models. It's the union of Hussain-McLaren-Stuttard and StatisticsCanada.
- pure2.pml — All ARIMA models (p d q)(P D Q) with p and q between 0 and 2, P and Q also between 0 and 2, d either 0 or 1, and D always equal to 1. Does not include mixed models (50 models).
- pure3.pml — Same as pure2 but with p and q varying from 0

to 3 (70 models).
- pure4.pml and pure5.pml Analogue (90 and 110 models, respectively).
- st-pure2.pml and st-pure3.pml Same as pure2 and pure3, respectively, but containing only stationary models (d = 0).
- int-pure2.pml and int-pure3.pml Same as pure2 and pure3, respectively, but containing only integrated models (d = 1).
- mixed2.pml and mixed3.pml Same as pure2 and pure3, respectively, but including mixed models (162 models and 288 models, respectively).
- ARIMA.pml ARIMA models with no seasonal ARIMA part; all models from (0 0 0) to (3 1 3).

To use one of these files, include the section-key-value triple 'pickmdl','file','ONS.pml' (as an example) in your x13spec command.

You can also use your own model definition files. Your file must have the .pml extension and must be in the current directory, or you must provide the full path.

If the pickmdl section is set but no file name is provided by the user, the toolbox will use pure3.pml.

Remark 10: the fixedseas and camplet sections ----------
x13spec also accommodates two sections that have no meaning for the x13as program. These sections are 'fixedseas' and 'camplet'. The contents of these sections are not transmitted to x13as. Instead, they are passed to separate Matlab programs (fixedseas.m and camplet.m). Specifying the 'fixedseas' and/or the 'camplet' section together with the 'x11' or the 'seats' section allows you to perform two/three types of seasonal adjustments in one go.

fixedseas computes a trend and seasonal adjustment using a much simpler method than X-13ARIMA-SEATS. The results are embedded into the x13series object as variables 'tr' (for trend), 'sf' (for seasonal factor), 'sa' (for seasonally adjusted), and 'ir' (for irregular). fixedseas is much less successful in removing seasonality that X-13ARIMA-SEATS is, but it has the advantage of producing seasonal factors that do not change over time. It is also computationally much cheaper, and works with arbitrary frequencies.

The 'fixedseas' section supports the following keys:
- 'period' This is a single positive integer or a vector of positive integers. It determines the frequencies that are filtered out. If this key is not given, it is set equal to obj.period (i.e. typically 4 or 12).
- 'transform' fixedseas does an additive or a multiplicative (log-additive) decomposition of the data. You can specify here which one to use. If this argument is omitted, the decomposition is log-additive if obj.isLog is true and additive otherwise.

- 'type'   Determines how the trend is computed. Default is 'ma' for
              moving averages. Alternatives are 'hp' (for Hodrick-Prescott),
              'detrend' (using Matlab's detrend function), 'spline', and
              'polynomial'.
- 'typearg'   Additional arguments for 'type' can be specified here. For
              'hp', 'spline', and 'polynomial', see help fixedseas for
              an explanation. With 'detrend', the additional argument
              must be a date or datevector, indicating where breaks in
              the trend should be allowed.
- 'extend'   X13ARIMA-SEATS is used to estimate an ARIMA and compute
              forecasts and backcasts before fixedseas is used. These
              fore- and backcasts are appended and preappended,
              respectively, to the data, before computing the trend. This
              avoids problems of estimating the trend close to the edge
              of the sample. This cannot be achieved automatically by
              using fixedseats.m directly. It is, however, automatically
              done when the x13spec induces x13as to estimate an ARIMA
              model (either with an 'arima' or 'pickmdl' oder 'automdl'
              command). By default, 'extend' is set to 3*max(period), but
              you can amend this choice with the value following the
              'extend' key.

camplet computes a seasonal adjustment propsed by Abeln and Jacobs, 2015.
The results are embedded into the x13series object as variables 'csf'
(seasonal factor), 'csa' (seasonally adjusted), 'cer' (rolling forecast
error), and 'cg' (rolling trend estimate). camplet is less successful in
removing seasonality than X-13ARIMA-SEATS is, but it has the advantage
that it is computationally much cheaper and works with arbitrary
frequencies. Also, unlike fixedseas, it does allow for seasonality that
is changing over time.

- 'period'   This is a single positive integer that determines the
              frequency that is filtered out. If this key is not given,
              it is set equal to obj.period (i.e. typically 4 or 12).
              Unlike fixedseas, camplet does not support a vector for
              'period', To perform multiple camplet filterings, run them
              sequentially, using the original data first, and then the
              output of the first filtering round (for the first frequency)
              as the input for filtering out the second frequency, and so
              on.
- 'transform' camplet does an additive or a multiplicative
              (log-additive) decomposition of the data. You can specify
              here which one to use. If this argument is omitted, the
              decomposition is log-additive if obj.isLog is true and
              additive otherwise.
The following parameters are the ones defining the CAMPLET algorithm, see
the working paper for explanations:
- 'CA'   CA parameter (Common Adjustment).
- 'M'   M parameter (Multiplier).

- 'P' P parameter (Pattern).
- 'LE' LE parameter (Limit to Error).
- 'T' T parameter (Times).
- 'INITYEARS' The number of years used to initialize the algorithm. The CAMPLET algorithms sets this to 3, but you can override this choice.

makespec

```
makespec produces x13 specification structures. It makes the use of
x13spec easier by providing quick access to meaningful specification
combinations.

Usage:
  spec = makespec(shortcut, [shortcut2, ...])
  spec = makespec([shortcut], [section, key, value], ...)
  spec = makespec([shortcut], [section, key, value], [spec1], ...)

Available shortcuts are:
  'DIAGNOSTIC'    produce ACF and spectra of the data; this is useful to
                  determine if the data is seasonal at all
  'ACF'           subset of 'DIAGNOSTIC' without spectra (for quarterly
                  data); saves (partial) auto-correlation functions
  'SPECTRUM'      save some spectra
  'STOCK'         Data is a stock variable. (This is relevant for the types of
                  calendar dummies.)
  'FLOW'          Data is a flow variable.
  'AUTO'          let program select additive vs multiplicative filtering
  'MULTIPLICATIVE'    force multiplicative filtering
  'ADDITIVE'      force additive filtering
  'ESTIMATE'      estimate ARIMA, even if no seasonal adjustment is
                  computed
  'TRAMO'         use TRAMO to select model
  'TRAMOPURE'     use TRAMO, but do not consider mixed models
  'PICKFIRST'     use Census X-11 procedure to select model; pick the first
   or 'PICK'      that meets the criteria
  'PICKBEST'      use Census X-11 procedure to select model; check all
                  models and pick the best
  'CONSTANT'      adds a constant to the regARIMA model
  'AO'            allow additive outliers
  'LS'            allow level shifts
  'TC'            allow temporary changes
  'NO OUTLIERS'   do not detect outliers
  'TDAYS'         add trading day dummies to the regression and keep them
                  if they are significant
  'FORCETDAYS'    force seven trading day dummies on the regression
                  (even if not significant)
  'EASTER'        add an Easter dummy and keep it if significant
  'FCT'           compute forecast with default confidence bands
  'FCT50'         compute forecast with 50% confidence bands
  'X11'           compute Trend-Cycle and Seasonality using X-11
  'FULLX11'       same as X11, but save all available variables, except
                  intermediary iteration results
  'TOTALX11'      same as X11, but save all available variables,
                  including intermediary iteration results
  'SEATS'         compute Trend-Cycle and Seasonality using SEATS
```

```
  'FULLSEATS'     same as SEATS, but save all available variables
  'FIXEDSEASONAL' compute simple seasonal filtering with fixedseas
  'CAMPLET'       compute filtering with camplet algorithm
  'SLIDINGSPANS'  produces sliding span analysis to gauge the stability
                  of the estimation and filtered series
  'FULLSLIDINGSPANS'  same as SLIDINGSPANS, but save all available variables
  'HISTORY'       another stability analysis that computes the amount of
                  revisions that would have occurred in the past
  'FULLHISTORY'   same as HISTORY, but save all available variables

Moreover, makespec also accepts shortcuts that affect the tables that are
printed throughout. The default is to print only thise tables that are
explicity requested ('PRINTREQUESTED'). Alternatives are
  'PRINTNONE'     Do not print any tables (except some basic ones in series
                  or composite, respectively).
  'PRINTBRIEF'    Print a restricted set of tables.
  'PRINTDEFAULT'  Use the default as defined by the X-13 program.
  'PRINTALLTABLES' Print all tables, but no graphs.
  'PRINTALL'      Print all tables and graphs.

There are also meta-shortcuts:
  'DEFAULT' is equal to {'AUTO','TRAMOPURE','X11'  ,'AO','DIAG'}
  'X'       is equal to {'AUTO','PICKBEST' ,'X11'  ,'AO','DIAG'}
  'S'       is equal to {'AUTO','TRAMOPURE','SEATS','AO','DIAG'}
If no argument is ised (spec = makespec()), 'DEFAULT' is used.
You are free to add further meta-shortcuts according to your needs; see
the program text right at the beginning after the 'function' statement.

Note that shortcuts can be abbreviated but they are case sensitive; they
must be given in upper case letter. (This is so in order to distinguish
the shortcut 'X11' from the spec section 'x11', and shortcut 'SEATS' from
the spec section 'seats').

Multiple shortcuts can be combined, though some combinations are
non-sensical (such as X11 and SEATS, or TRAMO and PICK together).

No selection of shortcuts will ever accomodate all needs, unless the
shortcuts are as detailed as the original specification possibilities,
which would defy their intent. Therefore, one can also add normal
section-key-value triples as in x13spec (the second usage form above).
These settings are simply merged, working from left to right. This means
that later arguments overwrite earlier arguments.

So, makespec('NO OUTLIERS','AO') is the same as makespec('AO'), and in
makespec('AUTO','transform','function','none') the 'AUTO' shortcut is
overruled. Likewise, makespec('X','MULT') is the same as the 'X'
meta-shortcut, but forcing the logarithmic transformation of the data
('X' sets this to 'auto' and therefore lets x13as choose between no
transformation and the log).
```

You can also use an existing spec (created with makespec or with x13spec) as an argument in makespec (thirtd usage form above). The contents of this spec-variable will again be merged.

Example:

```
  spec = makespec('DIAG','AUTO','TRAMO','AO');
  x1 = x13(dates,data,makespec(spec, 'X11', ...
      'series','name','Using X11'));
  x2 = x13(dates,data,makespec(spec, 'SEATS', ...
      'series','name','Using SEATS');
  plot(x1,x2,'d11','s11','comb')
```

Most users will never use x13spec directly, but will always create their specs with makespec, because everything you can do with x13spec you can also do with makespec, plus you have the added convenience of the shortcuts (and even meta-shortcuts).

x13series.plot

```
plot (overloaded) plots the content of an x13series object

Usage:
  plot(obj)
  plot(obj, 'variable1', 'variable2', ...)
  plot(obj1, obj2, ..., 'variable1', 'variable2', ...)
  plot(h, obj, ...)
  plot(..., 'columnwise'|'rowwise'|'combined')
  plot(..., 'dateticks',['all','d','w','m','q','y','auto','matlab'])
  plot(..., 'dateticks','...', 'multdateticks', integer)
  plot(..., 'selection', boolean vector)
  plot(..., 'logscale')
  plot(..., 'normalized'|'meannormalized')
  plot(..., 'overlapperiods')
  plot(..., 'overlapyears')
  plot(..., 'span')
  plot(..., 'boxplot')
  plot(..., 'byperiod')
  plot(..., 'byperiodnomean')
  plot(..., 'separate')
  plot(..., 'options',{...})
  plot(..., 'quiet')
  [fh,ax] = plot(...)

The command can plot variables, ACF/PACF, and spectra contained in an
x13series object. It plots these types of items differently, and some of
the options apply only to some types of items.

The options can be abbreviated. However, at least four characters of the
option must be specified. Otherwise, the parameter is interpreted as the
name of an item that is to be plotted. For instance, in
plot(obj,'dat','log'), the program will try to plot the items 'dat' and
'log' (if available), but maybe you wanted to plot 'dat' on a log-scale.
To achieve that, you need to say (at the minimum) plot(obj,'dat,'logs')
[abbreviation of 'logscale'].

Inputs:
  obj         An x13series object.
  variable    The name of variables stored in obj. Default is 'dat'.
  h           Can be a figure handle or an axis handle. If it is an axes
              handle, then only one x13series and one variable can be
              specified, or the 'combined' keyword must also be used.
              This single variable of this single x13series is then
              plotted to the given axis.
  'rowwise'   The variables of an x13series are plotted in one row; each
              column contains the same variable of all x13sereies
              objects. This is the default.
```

'columnwise' This option swaps the location of the subaxes. With 'columnwise', the variables of an x13series are plotted in one column; each row contains the same variable of all x13series objects.

'combined' Plots all the requested information in one axis.

'dateticks' This is one of the following: 'all', 'd', 'w', 'm', 'q', 'y', 'matlab', 'auto', or 'default'. 'auto' makes a choice that often works. 'all' means that each datapoint on the dates-axis is labelled. 'matlab' means that Matlab's default is used. The default is 'auto'.

'multdateticks' Reduces the number of ticks. Example, if 'dateticks' is set to 'y' and 'multdateticks' is set to 3, then there is a tick at the beginning of every third year.

'selection' If a variable contains several time series (such as .fct, which conains the forecast as well as the lower and upper bounds of the confidence interval), then the vector following the 'selection' option determines which time series are plotted. For instance, plot(obj, 'fct', 'selection',[1 0 0]) plots only the forecast without the limits of the confidence band. Default is to plot all timeseries contained in an item. If the number of entries in the selection-vector does not match the number of time series contained in a variable, 'selection' is simply ignored.

'logscale' Applies only to variables (not ACF or spectra). Uses a logarithmic scale for the values.

'normalized' Applies only to variables. Normalizes data so that mean is zero and standard deviation is one. If 'normalized' and 'logscale' are used, the log of the data is first taken and the logarithmic data are then normalized.

'meannormalized' Applies only to variables. Same a 'normalized' but applies only to the mean.

'overlapperiods' The x-axis is either 1:12 (for monthly data) or 1:4 (for quarterly data). Each year's values are drawn as a separate line.

'overlapyears' The x-axis is one observation for each year. Each period (month or quarter) is drawn as a separate line.

'span' The x-axis is either 1:12 (for monthly data) or 1:4 (for quarterly data). Three lines are drawn, one containing the average for each month/quater over all years, and one showing the respective minimum or maximum.

'boxplot' Similar to 'overlapperiods', but instead of plotting each year as a line, here a boxplot for each month (quarter) is produced. You can use 'boxplot' and 'overlapperiods' together. This option requires the 'Statistics Toolbox'. If this Toolbox is not available, the option will be substituted by 'span'.

'byperiod' As with 'overlapperiods', 'span', and 'boxplot', the abscissa is either 1:12 or 1:4. For each period, the year-by-year

```
              development is depicted as a line, so for instance,
              plot(obj,'d10','byperiod') would show the development of
              the Jan, Feb, Mar erc seasonal factor from year to year.
              Also, for each month (quarter), the average factor is shown
              as a horizontal red line. The graph is similar to one of
              the more innovative plots provided by the Census Bureau
              plot utility.
  'byperiodnomean'   Same as 'byperiod', except that the average for each
              period is not shown in the graph.
  'separateperiods'  Same as 'byperiodnomean', but each month (quarter) gets
              its  own axis. Also the same as 'overlapyears', but with each
              line in its own subaxis.
  'options'   This overloaded plot method relies on Matlab's ordinary
              plot command to actually produce the figure. With
              'options' the user can specify any additional arguments
              that will be passed to the main plot function.
  'quiet'     Suppress warnings.

Outputs:
  fh          A handle to the figure that is created.
  ax          An array of handles to the individual axes that are
              contained in the figure.

Examples:

Straigtforward examples:
    plot(obj);
    plot(obj,'d10','d13');
    plot(obj,'dat','d11','combined');
    plot(obj1, obj2, 'd12','combined');

A more elaborate example:
    figure;
    ah = subplot(2,2,[1 3]);
    plot(ah,x,'dat','options',{'LineWidth',1.0});
    hold on;
    plot(ah,x,'d12','options',{'Color',[1,0,0],'LineWidth',2.0});
    title(ah,'\bfdata and trend');

    ah = subplot(2,2,2);
    plot(ah,x,'d10');

    ah = subplot(2,2,4);
    plot(ah,x,'d10','boxplot');
    title(ah,'\bfdistribution of seasonal factors (d10)');
```

x13series.seasbreaks

```
seasbreaks (overloaded) produces a special plot showing potential seasonal breaks

Usage:
  seasbreaks(obj)
  seasbreaks(h, obj, ...)
  seasbreaks(h, obj, sf, si, ...)
  seasbreaks(..., plotoptions)
  [fh,ax] = seasbreaks(...)

The plot produces a chart with one axis for each month (quarter)
displaying the seasonal factors as lines and the SI ratios as markers.
Normally, the lines should be relatively close to the markers. If for one
month (quarter), the markers are all below the line, and then suddenly
above it (or vice versa), this indicates a break in the seasonal
structure. The function returns a handle to the figure and a matrix of
handles to the individual axes.

The program works only if
 - the X-11 seasonal factors have been computed and 'd8' and 'd10' have
   been saved, or
 - the SEATS seasonal factors have been computed and 's10' and 's13' have
   been saved, or
 - the FIXEDSEAS seasonal factors have been computed (sf and si).
For SEATS, a new variable will be computed. We call it s8. It is defined
as s8 = s10 + s13.

Alternatively, the variables that represent the seasonal factor and the SI
variable can also be added as arguments manually (see third line in 'Usage'
above).

Inputs:
  obj          An x13series object.
  h            An optional figure handle.
  sf           Name of variable (as string) containing a seasonal factor.
  si           Name of variable (as string) containing an SI variable.
  plotoptions  Any options passed on to x13series.plot.

Outputs:
  fh           A handle to the figure that is created.
  ax           Handles to the axes in the figure.
```

x13composite.plot

```
plot (overloaded) plots the content of an x13composite object

Usage:
  plot(obj)
  plot(obj, 'variable1', 'variable2', ...)
  plot(obj, 'variable1', 'variable2', ..., 'dropcomposite')
  plot(obj1, 'variable1', 'variable2', ..., option1, option2, ...)
  plot(h, obj, ...)
  [fh,ax] = plot(...)

Inputs:
  obj            A x13composite object.
  variable       The name of variables stored in the x13series contained in
                 obj.
  h              Can be a figure handle or an axes handle. If it is an axes
                 handle, then only one variable can be specified. If the
                 x13composite object contains multiple series (which
                 normally it would), then one must also set the 'combined'
                 option. The single variable of all x13series in obj are
                 then plotted to the given axes.
  dropcomposite  Do not plot the composite series.
  options        See help on x13series.plot for explanation.

Outputs:
  fh             A handle to the figure that is created.
  ax             An array of handles to the individual axes that are
                 contained in the figure.

x13composite.plot really functions like x13series.plot, where all series
contained in the composite object are passed as individual series to the
x13series.plot routine. For instance, if obj is a x13composite with five
series,

==========================================================================
 X13-ARIMA-SEATS composite object
..........................................................................
 List of series:
-> Y
 - C
 - I
 - G
 - NX
..........................................................................
 Time of run: 24-Jul-2015 09:18:02 (4.0 sec)
==========================================================================

then plot(obj, [...]) --- which calls x13composite.plot --- is exactly
```

the same as plot(objC,obj.I,obj.G,obj.NX,obj.Y, [...]) --- which calls x13series.plot.

x13composite.seasbreaks

```
seasbreaks (overloaded) produces special plots showing potential seasonal breaks

Usage:
  seasbreaks(obj)
  seasbreaks(..., 'dropcomposite')
  seasbreaks(..., options)
  fh = seasbreaks(...)

Inputs:
  obj            A x13composite object.
  dropcomposite  Do not plot the composite series.
  options        Any options passed on to x13series.plot.

Outputs:
  fh          An array of handles to the figures that are created.

x13composite.seasbreaks really functions like x13series.seasbreaks, where
all series contained in the composite object are passed as individual
series to the x13series.seasbreaks routine.
```

x13toxls

x13toxls exports an x13series variable into an Excel file.

Usage:
```
x13toxls(x,filename,['overwrite'])
```

x is a x13series object. filename is the name of the Excel workbook that will be created. If you add the switch 'overwrite', an existing Excel file with the same name will be overwritten.

There is no function for exporting x13composite objects to Excel, but you can use x13toxls to export individual series contained in an x13composite. For instance, if x is a x13composite with three series, x.country, x.north, x.south, then x13toxls(x.country,'country.xlsx') will write the content of x.country into an Excel file.

TakeDayOff

```
TakeDayOff helps you define special variables for regressions in seasonal
adjustment that indicate whether holidays are located such that people might
have an incentive to take a day off (i.e. on a Tuesday or Thursday).

Usage:
  spec = TakeDayOff(specialdays,filename,specialweekdays,fromYear,toYear)

specialday    This is a matrix with three columns and a maximum of five rows.
              The first column is the month, the second the day, and the
              third the length of the holiday. Default is [12 24 2],
              indicating Christmas (two days starting on Dec 24). If you also
              want to test for, say, 4th of July, use this: [12 24 2; 7 4 1].
filename      The name of the file created on hard drive that contains the
              user variables (and that are read by the x13as.exe program).
              Default is '_user.dat'
specialweekdays    List of two integers, indicating the weekdays to test for
              at the beginning of a special day and at the end of it. Default
              is [3 5], so that the program tests if the beginning is a
              Tuesday (3) and the end is a Thursday (5).
fromYear, toYear   The user variable is created for this interval of years.
              Default is from 1900 to 2200.
spec          A x13spec object to be integrated into your x13 run.

Example:
  userspec = TakeDayOff();
  disp(userspec);
This produces
================================================================================
 X-13/X-12 specification object
................................................................................
 - regression
    ├─ user : (Dec24Tue Dec25Thu)
    ├─ file : _user.dat
    ├─ format : datevalue
    ├─ usertype : (holiday holiday)
    ├─ aictest : user
    └─ save : hol

This spec can be used by saying:
  spec = makespec('DEFAULT',userspec);
  x = x13(dates,data,spec);
```

preadjustOnePeriod

preadjustOnePeriod replaces the seasonally unadjusted data for one month or one quarter, respectively, by the adjusted data, and then recomputes the seasonal adjustment.

Usage:

```
obj = preadjustOnePeriod(obj,p)
```

obj is a x13series (x13composites are not supported) that contains some seasonal adjustment. p is an integer between 1 and obj.period. p can also be a vector of such integers.

Example: Let obj be an x13series object with monthly periodicity and seasonal adjustment. Then, obj = preadjustOnePeriod(obj,12) will replace the December values with the seasonally adjusted data, and recompute the seasonal adjustment. The procedure also removes all outliers (such as ao or ls or hol) using addCDT.

Procedure: The first step is to use addCDT to remove the outliers. The second step is to use addASC to compute the additive seasonal contribution (asc). Then, the asc of the particual period(s) is removed from the data and the seasonal adjustement is computed again.

addCDT

addCDT removes outliers and holday corrections from dat and stores the result as a new variable called cdt ('corrected data').

Usage:

```
obj = addCDT(obj)
obj = addCDT(obj,'...');
obj = addCDT(obj,'...','...', ...);
```

obj must be a x13series object. (x13composites are not supported.) If only the obj is given as argument, then the following series (if present) are removed from the data: 'ls','ao','tc','hol','td'. Alternatively, the user can also provide a list of series to remove from the data, e.g. addCDT(obj,'ls') will remove only level shifts ('ls') from the data. The result is stored as obj.cdt.

addASC

addASC computes the absolute seasonal contribution in an x13series object and places it into the object as new variable called asc ('absolute seasonal contribution').

Usage:

```
obj = addASC(obj)
```

obj must be a x13series object with some form of seasonal adjustment (X11, SEATS, FIXEDSEAS, or CAMPLET). x13composites are not supported. The returned obj contains a new series, called obj.asc. This series contains the absolute seasonal contribution.

If the seasonal adjustment is additive, the absolute seasonal contribution is simply the seasonal factor (and there is not much point in applying addASC). If the seasonal adjustment is multiplicative, however, then asc = (sf-1)*tr, where sf is the multiplicative seasonal factor and tr is the trend component.

addspectrum

```
addspectrum computes the spectrum of a variable using the Signal Processing
Toolbox and adds the result to an x13series.

Usage:
  obj.addspectrum(v,d,vname,descr);

Inputs:
  obj     An x13series object.
  v       Variable contained in obj.
  d       Number of differences. d=0 means that the spectrum of the data
          itself is computed. Setting d=1 computes the spectrum of the first
          difference of the variable.
  vname   Name of the new variable that is created.
  descr   Short text describing the new variable.

Example: We assume that dates and data contain the dates and the observations
of a timeseries that will be seasonally adjusted.
  spec = makespec('ADDITIVE','FIXEDSEAS','CAMPLET');
  obj = x13([dates,data],spec);
  obj.addspectrum('sa' ,1,'sfa','Spectrum of fixed seasonal adjustment');
  obj.addspectrum('csa',1,'sca','Spectrum of camplet seasonal adjustment');
  plot(obj,'sfa','sca','combined');
```

addacf

```
addacf computes the autocorrelation function of a variable using the
Econometrics Toolbox and adds the result to an x13series.

Usage:
  obj.addacf(v,d,vname,descr);
  obj.addacf(v,d,vname,descr,nlags);

Inputs:
  obj     An x13series object.
  v       Variable contained in obj.
  d       Number of differences. d=0 means that the ACF of the data
          itself is computed. Setting d=1 computes the ACF of the first
          difference of the variable.
  vname   Name of the new variable that is created.
  descr   Short text describing the new variable.
  nlags   Number of lags to compute (default is 2*obj.period).

Example: We assume that dates and data contain the dates and the observations
of a timeseries that will be seasonally adjusted.
  spec = makespec('ADDITIVE','FIXEDSEAS');
  obj = x13([dates,data],spec);
  obj.addacf('ir' ,1,'fai','ACF of fixed irregular');
  plot(obj,'fai');
```

addpcf

```
addpcf computes the partial autocorrelation function of a variable using the
Econometrics Toolbox and adds the result to an x13series.

Usage:
  obj.addpcf(v,d,vname,descr);
  obj.addpcf(v,d,vname,descr,nlags);

Inputs:
  obj     An x13series object.
  v       Variable contained in obj.
  d       Number of differences. d=0 means that the PACF of the data
          itself is computed. Setting d=1 computes the PACF of the first
          difference of the variable.
  vname   Name of the new variable that is created.
  descr   Short text describing the new variable.
  nlags   Number of lags to compute (default is 2*obj.period).

Example: We assume that dates and data contain the dates and the observations
of a timeseries that will be seasonally adjusted.
  spec = makespec('ADDITIVE','FIXEDSEAS');
  obj = x13([dates,data],spec);
  obj.addpcf('ir' ,1,'fpi','PACF of fixed irregular');
  plot(obj,'fpi');
```

trendfilter

```
trendfilter produces a smoothed version of the data.

Usage:
  tr = trendfilter(data)
  tr = trendfilter(data,[method])
  tr = trendfilter(data,[method,parameters])
  tr = trendfilter(data,['mirror'|'extend', number])

data      An array. Each column is a time series and is smoothed separately.

method    Method used to smooth, possibly followed by one r several
          parameters. Possibilities are:
  'mean'                       Artithmetic mean over whole column.
  'deviation'                  Deviation of column means from row means.
  'reldeviation'               Relative deviation of column means from row
                               means.
  'ma' or 'ma',p1,p2,...       A simple moving average, or a convolution of
                               simple movong averages.
  'cma'                        A centered moving average over a range of minus
  'cma',p1,p2,...              p1/2 lags to plus p1/2., or a convolution of
                               such moving averages.
  'spencer' or 'spencer15'     A special 15-term moving average.
  'henderson',t                The Henderson filter with t terms.
  'bongard',t                  The Bongard filter with t terms.
  'rehomme-ladiray',t,p,h      The Rehomme-Ladiray filter with t terms, which
                               does perfectly reproduce polynome of order n,
                               and minimized a weighted average of the
                               Henderson and the Bongard criteria (with h being
                               the weight of the Henderson criterion).
  'detrend'                    A linear trend is fitted to the data.
  'detrend',bp                 A continuous, piecewise linear trend is
                               fitted to the data. 'bp' is the (row) vector of
                               breakpoints.
  'hp',lambda                  For the Hodrick-Prescott filter, an
                               additional argument must be given. lambda
                               is a smoothing parameter lambda. The
                               greater lambda, the smoother the trend.
  'spline',roughness           Fits a smoothing cubic spline to the data.
                               'roughness' is a number between 0.0
                               (straight line) and 1.0 (no smoothing), see
                               doc csaps.
  'polynomial',degree          Fit a polynomial of specified degree to the
                               data, see doc polyfit.
  'epanechnikov',bandwidth     Fit a kernel regression through the data using
                               the Epanechnikov kernel function.

'mirror',p    The p first and the p last observations are mirrored and
```

pre-appended and appended to the data, respectively. This reduces edge of sample problems. The mirrored part of the trend is removed and not returned in tr.

'extend',p Same as 'mirror', but without switching the order. This method works well only with stationary data. If in doubt, use 'mirror' rather than 'extend'.

seasfilter

```
seasfilter splits data using splitperiods, smoothes them with trendfilter,
and joins them together again with joinperiods.

Usage:
  f = seasfilter(data,p)
  f = seasfilter(data,p,varargin)

data is a column vector or an array of column vactors containing the data.
p is the periodicity of the data (so for instance, if you work with monthly
observations, p would be 12).
Additional arguments can be given that are passed on to trendfilter.

seasfilter performs a smoothing of the data for each period separately (so for
the sequence of observations in Januar, in February, etc, separately). In the
contect of seasonal filtering, this procedure smoothes the SI-components
(difference between the data and the trend), which are then called seasonal
factors.

Example:
data = sin(0.75*pi*(1:120)/120)';
s = [-0.2 0 0 0.1 0.4 0.6 0.2 -0.4 -0.3 -0.5 0 0.3];
s = repmat(s',10,1);
r = randn(120,1);
noisy = data + s*0.5 + r*0.2;
tr = trendfilter(noisy,'epanech',25);
si = normalize(noisy,tr);
sf = seasfilter(si,12,'spline',0.02);
sa = normalize(data,sf);
figure('Position',[416 128 560 673]);
subplot(2,1,1); plot([data,tr,sa],'linewidth',1); grid on;
subplot(2,1,2); plot(sf,'linewidth',1); grid on;
```

kernelweights

kernelweights returns a vector that can be used with wmean or conv to smooth a time series.

Usage:

```
w = kernelweights('ma',p1[,p2,p3,...])
w = kernelweights('cma',p1[,p2,p3,...])
w = kernelweights('spencer',[p])
w = kernelweights('henderson',p1[,p2,p3,...])
w = kernelweights('bongard',p1[,p2,p3,...])
w = kernelweights('rehomme-ladiray',t,p,h[,t2,p2,h2,....])
w = kernelweights(kernel of group 1,b[,b2,b3,...])
w = kernelweights(kernel of group 2,b,l[,b2,l2,...])
```

w is a vector of weights. If used on an array of data together with conv or wmean, it returns a smoothed version of the data.

'ma' or 'ma',p1,p2,...	A simple moving average, or a convolution of simple movong averages.
'cma'	A centered moving average over a range of minus
'cma',p1,p2,...	p1/2 lags to plus p1/2., or a convolution of such moving averages.
'spencer' or 'spencer15'	A special 15-term moving average.
'henderson',t	The Henderson filter with t terms.
'bongard',t	The Bongard filter with t terms.
'rehomme-ladiray',t,p,h	The Rehomme-Ladiray filter with t terms, which does perfectly reproduce polynome of order n, and minimized a weighted average of the Henderson and the Bongard criteria (with h being the weight of the Henderson criterion).

Group 1 (kernels with finite support):
'uniform','triangle','biweight' or 'quartic','triweight','tricube', 'epanechnikov', 'cosine','optcosine','cauchy'.

Group 2 (kernels with infinite support):
'logistic','sigmoid','gaussian' or 'normal','exponential','silverman'.

Kernels from group 1 are followed by a single parameter indicating the bandwidth. Kernels from group 2 have infinite support, and even Matlab cannot return an infinite vector. Of only one parameter is given, kernelweigths will return a vector that is long enough to contain all weights that are at least 1e-15. In a second argument is provided, this second argument is the length of the vector that is returned.

Example:
Let data be an array of noisy data:

```
data1 = (1:100)/100;
data2 = sin(2*pi*(1:100)/100);
data = [data1;data2]';
r = randn(101,2); r = r(2:end,:)*0.2 + r(1:end-1,:)*0.05; % autocorr noise
```

```
noisy = data + r;
w = kernelweights('epanech',20);
s = wmean(noisy,w);
figure('Position',[440 96 560 761]);
subplot(2,1,1); plot([noisy(:,1),s(:,1),data(:,1)],'linewidth',1); grid on;
subplot(2,1,2); plot([noisy(:,2),s(:,2),data(:,2)],'linewidth',1); grid on;
```

wmean

```
wmean computes a weighted mean.

wmean.m provides the same functionality as conv.m (a convolution), with
the following differences:
- The treatment at the edge of the sample is different. Only those weights
  are used that correspond to existing data.
- The sum of the used weights is normalized to one.
- wmean returns the same number of components as data has.

Usage: s = wmean(data,w)

  data    An array of data, organized columnwise.
  w       a vector with an odd number of values.
  s       s is the convolution of data and w.

So s(t) = s(t-b:t+b)*w, if the length of w is 2b+1. If some of the data are
outside the support, they are truncated for the computation. Moreover, the
result s(s) is divided by the sum of the weights in w that are used (i.e. not
truncated), thereby ensuring that the result is a proper weighted mean.

Example: Let data be some column vector containing a timeseries. Then,
  s = wmean(data,[1 1 1 1 1])
returns a moving average of data with a bandwidth of 5.
  load unemp;
  plot(unemp.dates,[unemp.data,wmean(unemp.data,[1 1 1 1 1])]);
```

splitperiods

splitperiods splits one vector of data into several columns, each containing a particular seasonal component.

Usage: sdata = splitperiods(data,p)

```
data      A column vector containing a time series.
p         The period of observation of the data.
sdata     an array with p columns, each containing a part of data.
```

Example: Let data contain monthly observations of some variable. Here, we use the US civilian unemployment rate:

```
load unemp
sdata = splitperiods(unemp.data,12);
```

Now sdata contains 12 columns. The first contains all observations from January, the second from February, etc.

```
nanmean(sdata) returns

Columns 1 through 7
  6.8769    6.7615    6.5231    6.0436    6.0436    6.4718    6.4359
Columns 8 through 12
  6.2026    6.0763    5.9579    6.0105    6.0158
```

These are the average unemployment rate over the whole sample of years, separately for each month.

joinperiods

joinperiods is the opposite of splitperiods.

Example:

```
load unemp; d = unemp.data; n = numel(d);
m = splitperiods(d,12);
plot(m,'linewidth',1); grid on;
legend('Jan','Feb','Mar','Apr','May','Jun','Jul','Aug','Sep','Oct', ...
    'Nov','Dec');
```

fillholes

fillholes filles missing values of a dataarray columnwise with linear interpolations. If data are missing at the edge of the vector, the missing values are left untouched, that is, no extrapolations are performed.

Usage: data = fillholes(data)

x11

x11 computes a simplified X-11 seasonal adjustment. It is a bit less refined than the Census Bureau original, but it has the advantage that it works with arbitrary frequencies, not just monthly or quarterly.

CAUTION: The program computes only a simplified version of the original X-11 algorithm. Most notably, there is no account of trading day corrections, and extreme values are also not taken care of. Still, the adjustment is often quite good, the program is applicable to any seasonality (not just 4 and 12 as the US Census Bureau program), and the code of the program can be instructive to develop variations.
Later versions of X-11 used an estimated ARIMA model to produce fore- and backcasts in order to alleviate the edge of sample problem that offurs in any filtering using moving averages. This program does not use ARIMA, but instead 'mirrors' at the left and right and applies the filtering after that. This simple technique appears to get rid of the edge of sample problem rather well.

NOTE: This program does *not* use the original X-11 executable program from the US Census Bureau. It does not support many of the options of that program either. This program merely tries to replicate the most important steps of the seasonal adjustment performed by the X-11 algorithm using Matlab directly. In other words, this is a Matlab implementation of a somewhat simplified version of the X-11 algorithm.
This fact also implies that, unlike the U.S: Census software, this implementation accommodates arbitrary frequencies, not just monthly or quarterly. This program is just a small addition to the toolbox that makes it more complete.

Usage:

```
  s = x11(data,period);
  s = x11([dates,data],period);
  s = x11(... ,transform);

  s   This is a structure containing the following components:
      s.p     = period
      s.type  = type of decomposition
      s.dates = dates vector
      s.dat   = data vector
      s.d13   = trend
      s.d11   = seasonally adjusted data
      s.d10   = seasonal factor (cycle)
      s.d13   = irregular component
              .
              .
              .
      The other components are from intermediate computation steps. Their
      meaning can be taken from the documentation of x13as.exe.
```

```
transform  Must be one of the following: 'additive','none','multiplicative',
      'logadditive','pseudoadditive'. It indicates the type of decomposition.
      'additive' or 'none' : data = tr + sf + ir, sa = tr + ir.
          'multiplocative' : data = tr * sf * ir, sa = tr * ir.
             'logadditive' : log(data) = log(tr*sf*ir), sa = exp(tr+ir).
          'pseudoadditive' : data = tr * (sf + ir - 1), sa = tr * ir.
```

seas

```
seas is a simple program that demonstrates the use of the programs in the
seas directory of the X-13 toolbox. These programs are:

trendfilter.m         Computes a trend (i.e. smoothed) version of the data.
                      You can choose from many different methods to do that.
seasfilter.m          Splits data using splitperiods, smoothes them with
                      trendfilter, and joins them together again with
                      joinperiods.
normalize.m           Computes the difference or the ratio, respectively, of
                      two series, depending on whether the decomposition is
                      additive or multiplicative, respectively.
splitperiods.m        Splits the data into their periods. For instance, with
                      monthly data, split periods makes twelve times series
                      out of your data, one for each month of the year.
joinperiods.m         Reverse of splitperiods.
fillholes.m           Linear interpolation of missing values.
wmean.m               Computes the weighted mean. Similar to Matlab's conv
                      command, but with smarter treatment of the edge of the
                      data.
kernelweights.m       Computes the weights for a wide range of kernels. Used
                      by trendfilter.m and seasfilter.m in conjunction with
                      wmean.m
fixedseas.m           A rather elaborate that produces a rather simple version
                      of seasonal adjustment in which the seasonal factors are
                      kept fixed over the years.
x11.m                 An simplementation of a much simplified version of the
                      original X-11 method of the U.S. Census Bureau.
camplet.m             A form of seasonal adjustment that was recently
                      developed and that does not produce revisions when data
                      are added to the time series. It does that because the
                      smoothing is completely backward looking (so no centered
                      filters at all). camplet is separately implemented and
                      does not use the other tools provided here.
seas.m                This file. You can experiment with the implementation
                      in this file, and develop your own seasonal adjustment
                      routine starting from seas.m.

Usage of seas.m
  s = seas(data,p)
  s = seas(data,p,type)
  [s,x] = seas(...)

data is either a column vector or an array with two columns. In that case,
the left column is a date vactor and the right column is the data vector.

p is the period of the data that is to be filtered out. So, typically, with
monthly data, for instance, p should be set to 12.
```

type is either 'additive' or 'multiplicative'. 'additive' implies that the seasonal factor will be zero on average and are subtracted from the unadjusted data to get to the seasonally adjusted data. If type is 'multiplicative' the seasonal factor is one on average, and the data is divided by the seasonal factor to get the seasonally adjusted data.

s contains the output in a struct, x contains that information but packed into an x13series object.

fixedseas

```
fixedseas computes a simple seasonal filter with fixed seasonal factors.

Usage:
  s = fixedseas(data,period);
  s = fixedseas([dates,data],period);
  s = fixedseas(... ,transform);
  s = fixedseas(... ,method);
  s = fixedseas(... ,method,methodarg);
  [s,aggr] = fixedseas(...);

data must be a vector. fixedseas is NaN tolerant, meaning data can contain
NaNs.

period is a positive number which indicates the length of the seasonal
cycle (i.e. period = 12 for monthly data, period = 7 for daily data
having a weekly cycle, or period = 5 if the data is weekdaily).

The optional arguments determine if the filtering should be done
additively or multiplicatively, and the method of filter to use for
computing the trend.

'transform' is one of the following:
  'none' or 'additive'        The decomposition is done additively. This
                              is the default.
  'logadditive'               The log is applied to the data, the
                              decomposition is then applied additively, and
                              the exponential of the result is returned.
  'multiplicative'            The decomposition is done multiplicatively.

'method' (and 'methodarg') determines the method of trend. For possible
choices, see help trendfilter. Default is a centered moving average with
length equal to period ('cma',period).

'period' can also be a positive vector. In that case, the seasonal filtering
is performed several times, removing cycles at all desired frequencies. In
that case, 'transfrom' and 'method' can be cellarrays, containing one method
(plus argument) for each period. The returned s is then a structure with as
many components as there are components in 'period'.

If period is a vector and transfrom is the same for each period, an additional
aggregated structure is appended to s, providing the cumulated seasonal
factors etc. In that case, aggr is returned as true.

The program will select default values for 'lambda','roughness', or
'degree', respectively, if you do not specify them. If you use a vector
for the 'period' argument (filtering out multiple periods), then you can
also specify vectors of lambda/roughness/degree-arguments, one for each
```

```
component of your period-vector.

s is a struct with the following fields:
  .period     Period(s) that has/have been filtered.
  .transform  Either 'none' or 'log' or 'mult'.
  .method     The method used for computing the trend.
  .methodarg  possibly a parameter for the smoothing algorithm.
  .dates      The original dates. If none were provided, this is just a vecor
              counting from 1 to the number of data points.
  .dat        The original data.
  .tr         Long term trend (by default the moving average, but other
              choices are possible, see above).
  .sa         Seasonally adjusted series (= dat – sf, or exp(dat – sf),
              respectively).
  .sf         Seasonal factors.
  .ir         Irregular (= sa – tr or exp(sa – tr), respectively).

Data is decomposed into the three components, trend (tr), seasonal factor
(sf), and irregular (ir). For the additive decomposition, it is always
the case that data = tr + sf + ir. Furthermore, sa = data - sf (or
equivalently, sa = tr + ir). For the multiplicative decomposition, data =
tr * sf * ir, and sa = data ./ sf (or equivalently, sa = tr * ir).

Example 1:
  truetrend = 0.02*(1:200)' + 5;
  % truecycle = sin((1:200)'*(2*pi)/20);
  truecycle = repmat([zeros(7,1);-0.6;zeros(11,1);0.9],ceil(200/20),1);
  truecycle = truecycle(1:200);
  truecycle = truecycle - mean(truecycle);
  trueresid = 0.2*randn(200,1);
  data = truetrend + truecycle + trueresid;
  s = fixedseas(data,20);
  figure('Position',[78 183 505 679]);
  subplot(3,1,1); plot([s.dat,s.sa,s.tr,truetrend]); grid on;
  title('unadjusted and seasonally adjusted data, estimated and true trend')
  subplot(3,1,2); plot([s.sf,truecycle]); grid on;
  title('estimated and true seasonal factor')
  subplot(3,1,3); plot([s.ir,trueresid]); grid on;
  title('estimated and true irregular')
  legend('estimated','true values');

Example 2 (multiple cycles):
  truecycle2 = 0.7 * sin((1:200)'*(2*pi)/14);
  data = truetrend + truecycle + truecycle2 + trueresid;
  s = fixedseas(data,[14,20],'hp');
  figure('Position',[78 183 505 679]);
  subplot(3,1,1); plot([s.dat,s.sa,s.tr,truetrend]); grid on;
  title('unadjusted and seasonally adjusted data, estimated and true trend')
  subplot(3,1,2); plot([s.sf,truecycle+truecycle2]); grid on;
```

```
  title('estimated and true seasonal factor')
  subplot(3,1,3); plot([s.ir,trueresid]); grid on;
  title('estimated and true irregular')
  legend('estimated','true values');
```

Note that fixedseas(data,[14,20]) is not the same as fixedseas(data,[20,14]). The filters are applied iteratively, from left to right. The ordering matters, so the results differ.

Detailed description of the model: Let x be some timeseries. As an example, we compute fixedseas(x,6).

*** STEP 1 ***

We compute a 6-period centered moving average,

```
  trend(t) = sum(0.5x(t-3)+x(t-2)+x(t-1)+x(t)+x(t+1)+x(t+2)+0.5x(t+3))/6
```

The weights on the extreme values of the window are adapted so that the sum of the weights is equal to period. So, for instance, if period = 7, the weight on x(t-3) and x(t+3) would be 1.0; if period = 6.5, the weight would be 0.75.

[Note: By default the trend is computed as the centered moving average, and this is what is explained here. Other specifications are possible, namely detrend, hodrick-prescott, spline, and polynomial.]

*** STEP 2 ***

Compute the individual deviations of x from the trend,

```
  d = x - trend.
```

*** STEP 3 ***

Compute the average deviation over all observations on a cycle of 6 periods,

```
  m(1) = mean(d(1) + d(7) + d(13) + d(19) + ...)
  m(2) = mean(d(2) + d(8) + d(14) + d(20) + ...)
  ...
  m(6) = mean(d(6) + d(12) + d(18) + d(24) + ...)
```

*** STEP 4 ***

Normalize m so that its average is zero,

```
  n = (m(1)+m(2)+...+m(6))/6
  sf(1) = m(1) - n, sf(2) = m(2) - n, ..., sf(6) = m(6) - n
```

These are the seasonal factors.

*** STEP 5 ***

Compute the seasonally adjusted time series as sa = x - sf.

*** STEP 6 ***

Compute the irregular as ir = sa - trend. This is the part of the fluctuations of x that is not explained by the seasonal factors or the trend (= moving average).

STEP 1 as described here is for the 'moving averqage' trend type, which is the default. This step is different for the different trend types that are available. STEP 2 to 6 are, however, independent of the type of trend that is computed.

If the multiplicative option is used, the logarithm of the data is

processed and the exponential of the processed time series is returned. So, s = fixedseas(data,period,'log') is materially the same as s2 = fixedseas(log(data),period). Then, exp(s2.sa) = s.sa, exp(s2.sf) = s.sf, and exp(s2.tr) = s.tr.

camplet

```
camplet computes the Camplet seasonal adjustment.

Source: Barend Abeln and Jan P.A.M. Jacobs, "Seasonal adjustment with and
without revisions: A comparison of X-13ARIMA-SEATS and camplet," CAMA Working
Paper 25/2015, Australian National University, July 2015.

Note: The initialization algorithm is different from the one proposed by the
authors. As a result, the adjustment of the first few years of data is
different then when using the authors' original algorithm. Moreover, on very
volatile time series, these differences remain throughout the sample because
the automatic parameter adjustments are not identical. In practice, the
differences should be rather small.

Usage:
  s = camplet(data,period);
  s = camplet([dates,data],period);
  s = camplet(... ,'log');
  s = camplet(... ,'verbose');
  s = camplet(... , name,value, [name,value], ...);

data must be a vector. camplet is NaN tolerant, meaning data can contain
NaNs.

period is a positive number which indicates the length of the seasonal
cycle (i.e. period = 12 for monthly data, period = 7 for daily data
having a weekly cycle, or period = 5 if the data is weekdaily).

Some arguments are added as single keywords:
'additive' or 'none'          Implies that the analysis is performed on the
                              data as presented.
'multiplicative' or 'log'     The log of the data is first taken. After the
                              application of the algorithm, the exponential of
                              the result is returned. This amounts to a
                              multiplicative seasonal adjustment.
'verbose'                     During execution the program outputs detailed
                              information to the console whenever something
                              unusual happens (detection of an outlier or
                              pattern shift, for instance).

Other optional arguments are entered as name-value pairs. Possible names and
their meaning are:
'INITYEARS'   The number of years used to initialize the alorithm. Default
              is 3. (initT = INITYEARS * period is the number of observattions
              used for the initialization.)
'INITMETHOD'  The argument following this must be one of the following:
     ...,'mean' Takes the average daviation of the data from its mean over the
              interval 1:initT. The initial estimate of the seasonal factors
```

```
              is then the average of these deviations for each month/quarter.
     ...,'ma' Computes the deviation of the data (1:initT) from a centered
              moving average over period observations, instead.
     ...,'ls' ls stands for least squares. This option estimates a linear
              regression of the data (1:initT) on a constant and a linear
              trend. The average residuals per month/quarter are the starting
              values for the seasonal factor. The slope of this regression is
              the initial estimat of g. This method is the default.
'CA'          Initial CA parameter (Common Adjustment).
'M'           Initial M parameter (Multiplier).
'P'           Reset value for CA when pattern shift is detected.
'LE'          Initial LE parameter (Limit to Error).
'T'           Initial T parameter (Number of repetition before pattern shift
              is detected).
'LEshare'     Limit of outliers that invokes the 'volatile series' adjustment.
'CAadd'       Increment to CA parameter for volatile series.
'LEadd'       Increment to LE parameter for volatile series.
'LEmax'       Maximum limit to error.
'TIadd'       Increment of T parameter for volatile series when LE exceed
              LEmax.
'MUsub'       Reduction of MU parameter for volatile series when LE exceed
              LEmax.
'SIM'         Strictly between 0 and 1. The parameter determines the required
              similarity between consecutive errors to trigger a pattern
              shift.

s is a struct with the following fields:
  .dat        The original data.
  .dates      The original dates. If none were provided, this is just a vecor
              counting from 1 to the number of data points.
  .period     Period that has been filtered.
  .transform  Either 'none' or 'log'.
  .opt        Structure containing the selected parameters.
  .sa         Seasonally adjusted series.
  .sf         Seasonal factors.
  .fcst       Running forecast.
  .err        Running forecast error.
  .g          Running estimate of trend.
  .outlier    Number of consecutive outliers in a particular month/quarter.
  .pshift     Boolean indicating detection of a pattern shift.
  .currca     Changing value of CA.
  .ca         Changing value of CA.
  .m          Changing value of M.
  .le         Changing value of LE.
  .t          Changing value of T.

s.opt is a struct with the the parameters chosen by the user (or the default
parameters if nothing was selected): .INITMETHOD .INITYERAS .CA .M .P .LE .T
.LEshare .CAadd .LEadd .LEmax .TIadd .MUsub .SIM
```

```
Examples:
We assume that data is a column vector of data (the original time series)
with a quarterly frequency, and dates is an equally long vector containing
Matlab date codes.
  c = camplet(data,4);
  figure('Position',[440 160 560 700]);
  ah = subplot(2,1,1); plot(ah,[data,c.sa]); grid on;
  ah = subplot(2,1,2); plot(ah,c.sf,'k');    grid on;
If data is monthly, replace the 4 above by 12. Any other frequency is fine,
too, actually (for instance, with weekdaily data, searching for a weekday
pattern, use 5).
You can choose to perform a multiplicative filtering instead, and add detailed
feedback to the console on what is happening:
  c = camplet([dates,data],4,'verb','mult');
  figure('Position',[440 160 560 700]);
  ah = subplot(2,1,1); plot(ah,dates,[data,c.sa]); dateaxis('x'); grid on;
  ah = subplot(2,1,2); plot(ah,dates,c.sf,'k');    dateaxis('x'); grid on;
You can tweak the parameters. Here, we change the initial period and the
method of the initialization phase:
  c  = camplet([dates,data],4);
  c2 = camplet([dates,data],4,'INITMETHOD','ma','INITYEARS',5);
  plot(dates,[c.sf,c2.sf]); dateaxis('x'); grid on;
```

spr

```
spr computes the standard deviation of the residual from the seasonal
filtering done with fixedseas using different periodicities. It helps
identifying the periodicity of the cycle(s).

Usage:
  spr(data);
  [s,p,r,x] = spr(data);
  [s,p,r,x] = spr(data1, data1, ...);
  [s,p,r,x] = spr(..., options);

data or data1, data2, etc are individual vectors or data.
options are any options passed on to fixedseas. (For instance, 'add', or
'mult' or any other option of fixedseas.)

If used with no output variables, spr produces a graph showing the
standard deviation of the irregular of fixedseas with periods running
from one to a third the length of data. The spikes in this graph indicate
potential periods of cycles. One line is used for each data vecor given
as argument.

If used with output variables, s is the vector of standard deviations, p
is the vector 1:p, where p is a third of the length of data (so plot(p,s)
plots the spikes), and r is a matrix with all the residuals. x is a
boolean vector identifying spikes (extreme negative curvatures).

If multiple data are used, then s,p,r,x are cell vectors, containing the
results for each data vector separately.

Example:
  trend  = 0.02*(1:200)' + 5;
  cycle1 = 1.0 * sin((1:200)'*(2*pi)/14);
  cycle2 = 0.7 * sin((1:200)'*(2*pi)/20);
  resid  = 0.5 * randn(200,1);
  data   = trend + cycle1 + cycle2 + resid;
  % So we know that data has two periods, 14 and 20. We now try to find
  % these periods.
  figure; spr(data);
  % The graph reveals a clear spike at 14 (and an echo at 28 etc), which
  % we filter out now ...
  s = fixedseas(data,14);
  figure; spr(s.sa,s.ir);
  % The seasonally adjusted series and the residuals show no spike at 14
  % anymore, but a clear spike at 20 (and 40 and 60). We now take out
  % period 20 as well.
  s = fixedseas(data,[14,20]);
  figure; spr(s.sa,s.ir(:,end));
  % The seasonally adjusted series and the residuals show no spikes anymore.
```

DEMO for X13 Toolbox: Single Series Run

Preliminaries

```
% get correct path
p = fileparts(mfilename('fullpath'));   % directory of this m-file
% if the section is run with Shift-Ctrl-Enter ...
if isempty(p); p = [cd,'\']; end
% location of graphics files for use with X-13-Graph program
grloc = fullfile(p,'graphics\');

% size for figures with subplots
scSize = get(groot,'ScreenSize');   % size of physical monitor in pixels
scWidth = scSize(3); scHeight = scSize(4);
sizeFig = @(h,v) round([0.04*scWidth, scHeight*(1-0.08-v/100)-50, ...
    h/100*scWidth, v/100*scHeight]);

size1 = sizeFig(40,80);
size2 = sizeFig(80,45);
size3 = sizeFig(95,45);
size4 = sizeFig(70,70);
size6 = sizeFig(75,68);
size8 = sizeFig(95,65);
size9 = sizeFig(95,90);

% line width
lwidth = 78;

% single and double line
sline = repmat('-',1,lwidth+1);
dline = repmat('=',1,lwidth+1);

% display with wrapped lines and leading space
report = @(s) disp(WrapLines(s,lwidth,' '));

% write heading
```

```
clc; disp(dline);
report(['DEMONSTRATION OF X-13 TOOLBOX FOR MATLAB : ', ...
    'run on a single timeseries']);
report(['This script was developed with MATLAB Version ', ...
    '8.3.0.532 (R2014a)']);
disp(sline)
```

```
================================================================================
 DEMONSTRATION OF X-13 TOOLBOX FOR MATLAB : run on a single timeseries

 This script was developed with MATLAB Version 8.3.0.532 (R2014a)

--------------------------------------------------------------------------------
```

Loading Data

```
% US. Federal Highway Administration, Vehicle Miles Traveled
% [TRFVOLUSM227NFWA], retrieved from FRED, Federal Reserve Bank of St.
% Louis https://research.stlouisfed.org/fred2/series/TRFVOLUSM227NFWA/,
% December 31, 2014.

load(fullfile(p,'travel'));
report(['Source and discription of data: ',travel.source]);
name = 'miles traveled';

% travel   = fetchdata('TRFVOLUSM227NFWA', 'source','fred');
% travelSA = fetchdata('TRFVOLUSM227SFWA', 'source','fred');
% travel   = fetchdata('TRFVOLUSM227NFWA', 'source','fred', ...
%     'from',travelSA.dates(1));
```

```
 Source and discription of data: US. Federal Highway Administration, Vehicle
 Miles Traveled [TRFVOLUSM227NFWA], retrieved from FRED, Federal Reserve Bank
 of St. Louis https://research.stlouisfed.org/fred2/series/TRFVOLUSM227NFWA/,
 December 31, 2014.
```

Step 1: Quick and Dirty

```
disp(sline)
fprintf(' Step 1: ''Quick-and-Dirty''\n\n');

report(['We run a seasonal adjustment with the default parameters ', ...
    'and see what is coming out of it.']);

travel1 = x13([travel.dates,travel.data],makespec,'quiet');

disp(travel1.table('transform'));
report('CONCLUSION: The filtering will be additive.');

disp(travel1.table('d8a'));
report('CONCLUSION: The data are clearly seasonal.');
```

Step 1: 'Quick-and-Dirty'

We run a seasonal adjustment with the default parameters and see what is coming out of it.

Likelihood statistics for model fit to untransformed series.

Likelihood Statistics

Number of observations (nobs)	538
Effective number of observations (nefobs)	525
Number of parameters estimated (np)	3
Log likelihood (L)	-4911.1795
AIC	9828.3590
AICC (F-corrected-AIC)	9828.4051
Hannan Quinn	9833.3673
BIC	9841.1492

Likelihood statistics for model fit to log transformed series.

Likelihood Statistics

Number of observations (nobs)	538
Effective number of observations (nefobs)	525
Number of parameters estimated (np)	3
Log likelihood	1431.9971
Transformation Adjustment	-6344.8440
Adjusted Log likelihood (L)	-4912.8468
AIC	9831.6937
AICC (F-corrected-AIC)	9831.7398
Hannan Quinn	9836.7020
BIC	9844.4839

***** AICC (with aicdiff=-2.00) prefers no transformation *****
***** Additive seasonal adjustment will be performed. ****

CONCLUSION: The filtering will be additive.

D 8.A F-tests for seasonality

Test for the presence of seasonality assuming stability.

	Sum of Squares	Dgrs.of Freedom	Mean Square	F-Value
Between months	83474227519.5785	11	7588566138.14350	735.972**
Residual	5423554624.0395	526	10310940.34988	
Total	88897782143.6180	537		

**Seasonality present at the 0.1 per cent level.

Nonparametric Test for the Presence of Seasonality Assuming Stability

```
                Kruskal-Wallis      Degrees of    Probability
                  Statistic        Freedom         Level

                    506.3857          11            0.000%

           Seasonality present at the one percent level.

  Moving Seasonality Test
                        Sum of     Dgrs.of         Mean
                       Squares     Freedom        Square       F-value
   Between Years  1767547132.0747       43  41105747.257551       6.826**
           Error  2848372902.1406      473   6021930.025667

         **Moving seasonality present at the one percent level.

  COMBINED TEST FOR THE PRESENCE OF IDENTIFIABLE SEASONALITY

            IDENTIFIABLE SEASONALITY PRESENT

 CONCLUSION: The data are clearly seasonal.
```

Step 2: Identifying Model

```
disp(sline)
fprintf(' Step 2: Identifying model\n\n');

report(['''',travel1.arima,''' : PICKMDL has not found an ', ...
    'appropriate model. We will allow PICKMDL a wider menu and use ', ...
    'the mixed3.pml file containing 288 models. This will take a while...']);

spec1 = x13spec(makespec,'pickmdl','file','mixed3.pml');
travel1 = x13([travel.dates,travel.data],spec1,'quiet');

report(['''',travel1.arima,''' : PICKMDL has still not found an ', ...
    'appropriate model.']);

report('CONCLUSION: We will use TRAMO instead.');

spec2 = makespec('ADD','TRAMO','X11','AO','LS', 'DIAG', ...
    'series','name',name);
travel2 = x13([travel.dates,travel.data],spec2,'quiet');

report(['TRAMO finds a rather complicated mixed model: ', ...
    travel2.arima, '.']);

disp(travel2.table('regression'));

report(['CONCLUSION: The coefficients appear fine and the roots do ', ...
    'not indicate cancellation of terms. We can provisionally work ', ...
    'with this specification.']);
```

```
report(['However, MA(2) and MA(3) are not significant. We remove ', ...
    'them from the regression: (0 1 [1 4])(2 1 2).']);

spec2 = makespec(spec2,'automdl',[],[], ...
    'arima','model','(0 1 [1 4])(2 1 2)');
travel2 = x13([travel.dates,travel.data],spec2,'quiet');

disp(travel2.table('regression'));

report(['NOTE: Two outliers (a level shift and an additive outlier) ', ...
    'have been identified.']);

fh = figure('Position',size2);
plot(fh,travel2,'acf','pcf','comb');

disp(travel2.table('tukey'));
figure('Position',size3);
ax = subplot(1,3,1); plot(ax,travel2,'spr','str','comb');
ax = subplot(1,3,2); plot(ax,travel2,'sp1','st1','comb');
ax = subplot(1,3,3); plot(ax,travel2,'sp2','st2','comb');

disp(travel2.err);

report(['CONCLUSION: The ACF/PACF look good. The spectra also look ', ...
    'clean at the main frequencies, but do contain peaks at trading ', ...
    'day frequencies.']);
```

```
--------------------------------------------------------------------------------
 Step 2: Identifying model

 '(3 1 0)(0 1 1)' : PICKMDL has not found an appropriate model. We will allow
 PICKMDL a wider menu and use the mixed3.pml file containing 288 models. This
 will take a while...

 '' : PICKMDL has still not found an appropriate model.

 CONCLUSION: We will use TRAMO instead.

 TRAMO finds a rather complicated mixed model: (0 1 4)(2 1 2).

Estimation converged in    5 ARMA iterations,   64 function evaluations.

 Regression Model
 --------------------------------------------------------------------------------
                            Parameter        Standard
 Variable                    Estimate           Error      t-value
 --------------------------------------------------------------------------------
 Automatically Identified Outliers
   LS1979.May               -8510.4863      1698.76546        -5.01
   AO1995.Jan               16632.1144      1833.66019         9.07
 --------------------------------------------------------------------------------

 ARIMA Model:  (0 1 4)(2 1 2)
```

Nonseasonal differences: 1
Seasonal differences: 1

Parameter	Estimate	Standard Errors
Seasonal AR		
Lag 12	0.7485	0.09415
Lag 24	-0.4541	0.05328
Nonseasonal MA		
Lag 1	0.5296	0.04330
Lag 2	0.0446	0.04887
Lag 3	-0.0465	0.04852
Lag 4	0.1761	0.04281
Seasonal MA		
Lag 12	1.2779	0.09518
Lag 24	-0.5613	0.07722
Variance	0.58855E+07	
SE of Var	0.36326E+06	

Likelihood Statistics

Number of observations (nobs)	538
Effective number of observations (nefobs)	525
Number of parameters estimated (np)	11
Log likelihood (L)	-4842.2722
AIC	9706.5444
AICC (F-corrected-AIC)	9707.0590
Hannan Quinn	9724.9083
BIC	9753.4418

Roots of ARIMA Model

Root	Real	Imaginary	Modulus	Frequency
Seasonal AR				
Root 1	0.8241	1.2341	1.4840	0.1563
Root 2	0.8241	-1.2341	1.4840	-0.1563
Nonseasonal MA				
Root 1	1.2093	0.0000	1.2093	0.0000
Root 2	-1.7203	0.0000	1.7203	0.5000
Root 3	0.3875	1.6062	1.6523	0.2123
Root 4	0.3875	-1.6062	1.6523	-0.2123
Seasonal MA				
Root 1	1.1383	0.6970	1.3347	0.0874
Root 2	1.1383	-0.6970	1.3347	-0.0874

CONCLUSION: The coefficients appear fine and the roots do not indicate cancellation of terms. We can provisionally work with this specification.

However, MA(2) and MA(3) are not significant. We remove them from the regression: (0 1 [1 4])(2 1 2).

Estimation converged in 5 ARMA iterations, 54 function evaluations.

Regression Model

Variable	Parameter Estimate	Standard Error	t-value
Automatically Identified Outliers			
LS1979.May	-8488.3140	1699.49303	-4.99
AO1995.Jan	16525.4136	1840.47530	8.98

ARIMA Model: (0 1 [1 4])(2 1 2)
Nonseasonal differences: 1
Seasonal differences: 1

Parameter	Estimate	Standard Errors
Seasonal AR		
Lag 12	0.7486	0.09334
Lag 24	-0.4544	0.05313
Nonseasonal MA		
Lag 1	0.5434	0.03657
Lag 4	0.1600	0.03623
Seasonal MA		
Lag 12	1.2784	0.09446
Lag 24	-0.5617	0.07676
Variance	0.58970E+07	
SE of Var	0.36397E+06	

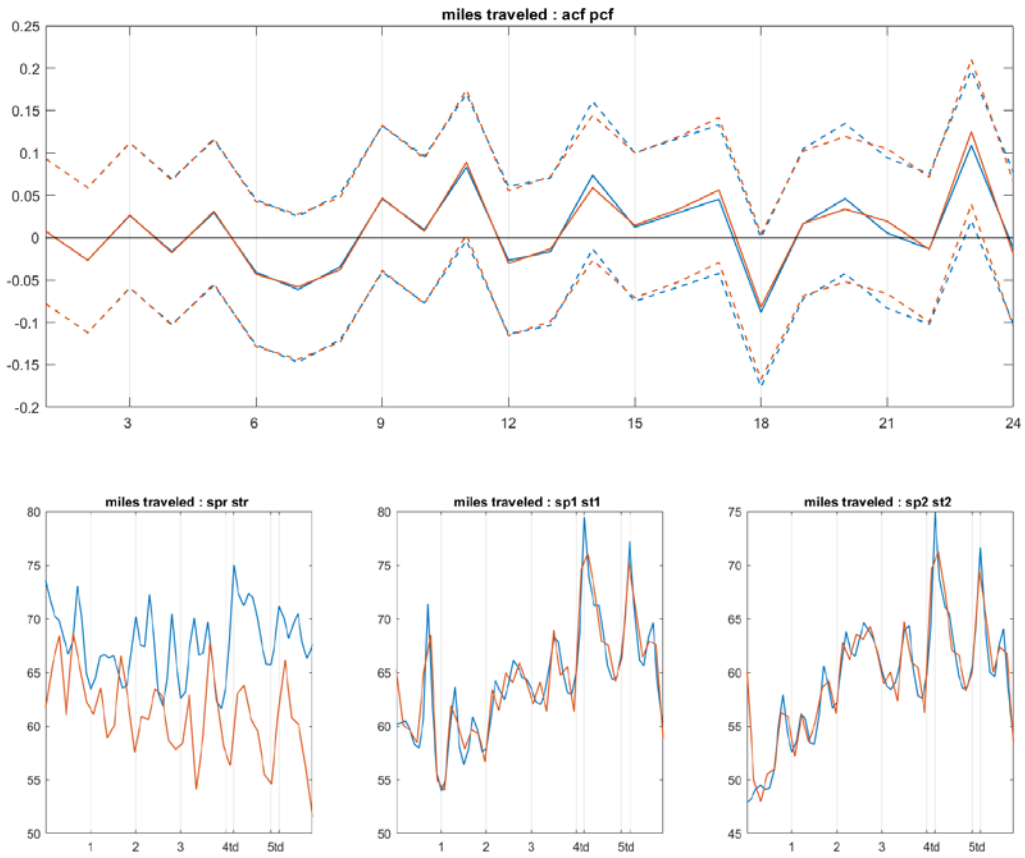
Likelihood Statistics

Number of observations (nobs)	538
Effective number of observations (nefobs)	525
Number of parameters estimated (np)	9
Log likelihood (L)	-4842.7910
AIC	9703.5819
AICC (F-corrected-AIC)	9703.9314
Hannan Quinn	9718.6069
BIC	9741.9525

Roots of ARIMA Model

Root	Real	Imaginary	Modulus	Frequency
Seasonal AR				
Root 1	0.8238	1.2338	1.4835	0.1563

```
   Root  2            0.8238    -1.2338     1.4835    -0.1563
  Nonseasonal MA
   Root  1            1.2096     0.0000     1.2096     0.0000
   Root  2           -1.8860     0.0000     1.8860     0.5000
   Root  3            0.3382     1.6200     1.6550     0.2172
   Root  4            0.3382    -1.6200     1.6550    -0.2172
  Seasonal MA
   Root  1            1.1380     0.6966     1.3343     0.0874
   Root  2            1.1380    -0.6966     1.3343    -0.0874
  -----------------------------------------------------------

 NOTE: Two outliers (a level shift and an additive outlier) have been
 identified.

Peak probabilities for Tukey spectrum estimator
  Spectrum estimated from 2006.Nov to 2014.Oct.

                                        S1       S2       S3       S4       S5       S6
TD
                                      ------   ------   ------   ------   ------   ------   ---
---
  Model Residuals                     0.309    0.108    0.217    0.233    0.088    0.249
0.740
  Prior Adjusted Series (Table B1)    0.960*   0.999**  0.998**  0.977*   0.999**  0.735
0.220
  Seasonally adjusted series (E2)     0.050    0.068    0.280    0.002    0.038    0.102
0.838
  Modified Irregular (E3)             0.130    0.067    0.261    0.002    0.031    0.111
0.860
  ----------
     ** - Peak Probability > 0.99,
      * - 0.90 < Peak Probability < 0.99

     Error messages generated from processing the X-13ARIMA-SEATS spec file
     C:\Users\Yvan\AppData\Local\Temp\X13\milesTraveled.spc:

 WARNING: At least one visually significant trading day peak has been
          found in one or more of the estimated spectra.

  Visually significant residual trading day peaks have been
  found in the spectral plots of the following series starting in 2006.Nov:

          differenced seasonally adjusted series (Table E2) (2 Trading Day peak(s))

          Modified irregular component (Table E3) (2 Trading Day peak(s))

 CONCLUSION: The ACF/PACF look good. The spectra also look clean at the main
 frequencies, but do contain peaks at trading day frequencies.
```



Step 3: Calendar dummies

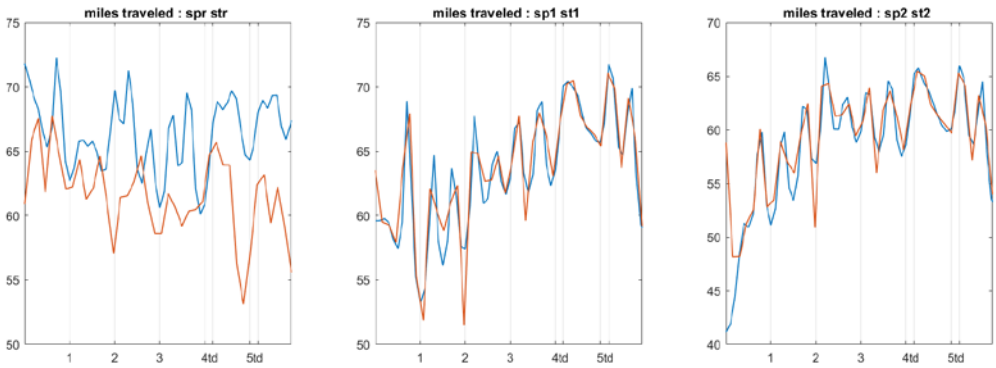
```
disp(sline)
fprintf(' Step 3: Calendar dummies\n\n');

spec3basic = x13spec(spec2,'automdl',[],[],'arima','model',travel2.arima);
spec3test = makespec(spec3basic,'EASTER','TD');

travel3test = x13([travel.dates,travel.data],spec3test,'quiet');

disp(travel3test.table('regr'));
report(['CONCLUSION: Trading day and Easter effects are significant. ', ...
    'We will keep them. Also, now three outliers are identified. ', ...
    'We will fix them as well.']);

spec3 = makespec(spec3basic, 'NO OUTLIERS', ...
    'regression','variables','(td easter[15] LS1979.May AO1993.May AO1995.Jan)', ...
    'regression','save','(td hol ao ls)');
travel3 = x13([travel.dates,travel.data],spec3,'quiet');

disp(travel3.table('tukey'));
figure('Position',size3);
ax = subplot(1,3,1); plot(ax,travel3,'spr','str','comb');
ax = subplot(1,3,2); plot(ax,travel3,'sp1','st1','comb');
```

```
ax = subplot(1,3,3); plot(ax,travel3,'sp2','st2','comb');

report('CONCLUSION: The spectra are clean now.');
```

```
--------------------------------------------------------------------------------
 Step 3: Calendar dummies

Estimation converged in   11 ARMA iterations,  146 function evaluations.

 Regression Model
 --------------------------------------------------------------------------------
                            Parameter        Standard
 Variable                    Estimate           Error      t-value
 --------------------------------------------------------------------------------
 Leap Year                   799.5865       559.56167         1.43

 Trading Day
   Mon                      -182.6238       163.87542        -1.11
   Tue                       210.8260       165.17994         1.28
   Wed                        33.7476       164.12109         0.21
   Thu                       563.1152       163.70578         3.44
   Fri                       586.1523       163.91323         3.58
   Sat                      -544.9328       164.31628        -3.32
  *Sun (derived)            -666.2845       164.29103        -4.06

 Easter[15]                  954.1090       348.07283         2.74

 Automatically Identified Outliers
   LS1979.May              -8286.0283      1670.00707        -4.96
   AO1993.May               7746.3405      1682.71670         4.60
   AO1995.Jan              16080.6408      1681.49657         9.56
 --------------------------------------------------------------------------------
  *For full trading-day and stable seasonal effects, the derived
   parameter estimate is obtained indirectly as minus the sum
   of the directly estimated parameters that define the effect.

 Chi-squared Tests for Groups of Regressors
 --------------------------------------------------------------------------------
 Regression Effect                  df      Chi-Square      P-Value
 --------------------------------------------------------------------------------
 Trading Day                         6           91.73         0.00
 Combined Trading Day and Leap Year Regressors
                                     7           93.92         0.00
 --------------------------------------------------------------------------------
 Trading Day                         6, 514           14.97         0.00

 ARIMA Model:  (0 1 [1 4])(2 1 2)
   Nonseasonal differences: 1
   Seasonal differences:    1
                                          Standard
 Parameter                   Estimate       Errors
 ---------------------------------------------------
 Seasonal AR
```

```
  Lag 12                       0.7859         0.09115
  Lag 24                      -0.3707         0.05555

 Nonseasonal MA
  Lag  1                       0.4750         0.03770
  Lag  4                       0.1715         0.03734

 Seasonal MA
  Lag 12                       1.3555         0.08750
  Lag 24                      -0.6089         0.06855

 Variance                     0.49280E+07
 SE of Var                    0.30416E+06
 ----------------------------------------------------

 Likelihood Statistics
 ---------------------------------------------------------------------
 Number of observations (nobs)                                    538
 Effective number of observations (nefobs)                        525
 Number of parameters estimated (np)                               18
 Log likelihood (L)                                         -4795.0362
 AIC                                                         9626.0724
 AICC (F-corrected-AIC)                                      9627.4242
 Hannan Quinn                                                9656.1224
 BIC                                                         9702.8135
 ---------------------------------------------------------------------

 Roots of ARIMA Model
  Root                  Real  Imaginary    Modulus  Frequency
  ------------------------------------------------------------
  Seasonal AR
   Root  1            1.0601     1.2545     1.6424     0.1383
   Root  2            1.0601    -1.2545     1.6424    -0.1383
  Nonseasonal MA
   Root  1            1.2430     0.0000     1.2430     0.0000
   Root  2           -1.8152     0.0000     1.8152     0.5000
   Root  3            0.2861     1.5817     1.6074     0.2215
   Root  4            0.2861    -1.5817     1.6074    -0.2215
  Seasonal MA
   Root  1            1.1131     0.6351     1.2816     0.0825
   Root  2            1.1131    -0.6351     1.2816    -0.0825
  ------------------------------------------------------------

 CONCLUSION: Trading day and Easter effects are significant. We will keep
 them. Also, now three outliers are identified. We will fix them as well.

Peak probabilities for Tukey spectrum estimator
  Spectrum estimated from 2006.Nov to 2014.Oct.

                                      S1       S2       S3       S4       S5       S6
TD
                                    ------   ------   ------   ------   ------   ------   ---
---
  Model Residuals                   0.392    0.109    0.309    0.289    0.223    0.311
```

```
0.724
  Prior Adjusted Series (Table B1)       0.960*   0.999**  0.998**  0.989*   1.000**  0.774
0.019
  Seasonally adjusted series (E2)        0.012    0.001    0.343    0.143    0.125    0.153
0.692
  Modified Irregular (E3)                0.163    0.001    0.353    0.141    0.130    0.173
0.751
  ----------
     ** - Peak Probability > 0.99,
      * - 0.90 < Peak Probability < 0.99

 CONCLUSION: The spectra are clean now.
```



More dummies...

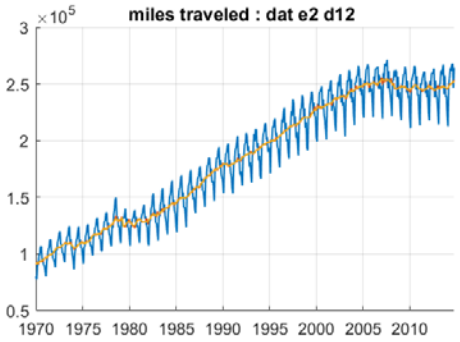
```
disp(' Trying more dummies...'); disp(' ');

s11 = makespec(spec3,'regression','variables','(labor[1] thank[1])');
s18 = makespec(spec3,'regression','variables','(labor[1] thank[8])');
s81 = makespec(spec3,'regression','variables','(labor[8] thank[1])');
s88 = makespec(spec3,'regression','variables','(labor[8] thank[8])');

t11 =  x13([travel.dates,travel.data],s11,'quiet');
t18 =  x13([travel.dates,travel.data],s18,'quiet');
t81 =  x13([travel.dates,travel.data],s81,'quiet');
t88 =  x13([travel.dates,travel.data],s88,'quiet');

report(['labor[8] and thank[1] or thank[8] are significant. The ', ...
    'labor[8] and thank[1] combination has a higher likelihood, so ', ...
    'we will include these dummies.']);

spec3 = s81;
travel3 = t81;
```

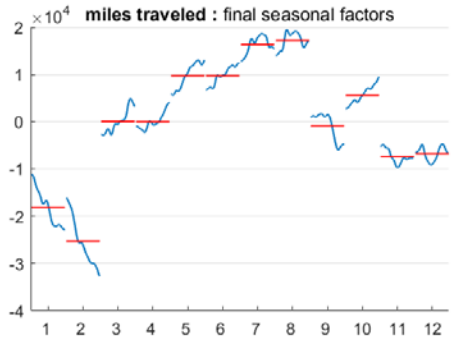
```
 Trying more dummies...

 labor[8] and thank[1] or thank[8] are significant. The labor[8] and thank[1]
 combination has a higher likelihood, so we will include these dummies.
```

Step 4: Do the Seasonal Filtering

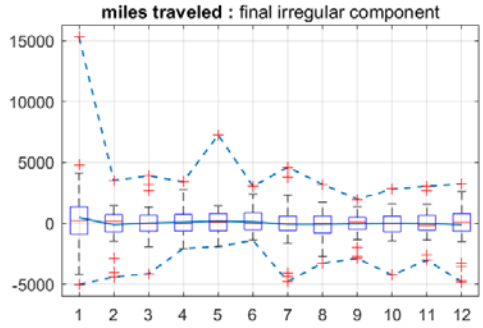
```
disp(sline)
fprintf(' Step 4: Performing the seasonal filtering\n\n');

spec4 = makespec(spec3,'X11');
travel4 = x13([travel.dates,travel.data],spec4,'quiet');

figure('Position',size4)
ax = subplot(2,2,1);
plot(ax,travel4,'dat','e2','d12','comb');
ax = subplot(2,2,2);
plot(ax,travel4,'d10','bymonth');
ax = subplot(2,2,3);
plot(ax,travel4,'spr','sp1','sp2','comb');
ax = subplot(2,2,4);
plot(ax,travel4,'d13','span','boxplot');

report(['The seasonal factors are rather stable (the graph on the ', ...
    'top right shows little variation). The decomposition (top left ', ...
    'graph) looks reasonable.']);

fh = figure('Position',size6);
seasbreaks(fh,travel4);

report(['A closer inspection into possible seasonal breaks reveals no ', ...
    'major problems either. This graph shows the seasonal factors and ', ...
    'the SI ratios separately for each month. We do see some quantitatively ', ...
    'important shifts, but they are all slow enough so that the X-11 ', ...
    'procedure can deal with it. We do not need to specify seasonal ', ...
    'breaks in the estimation.']);
```

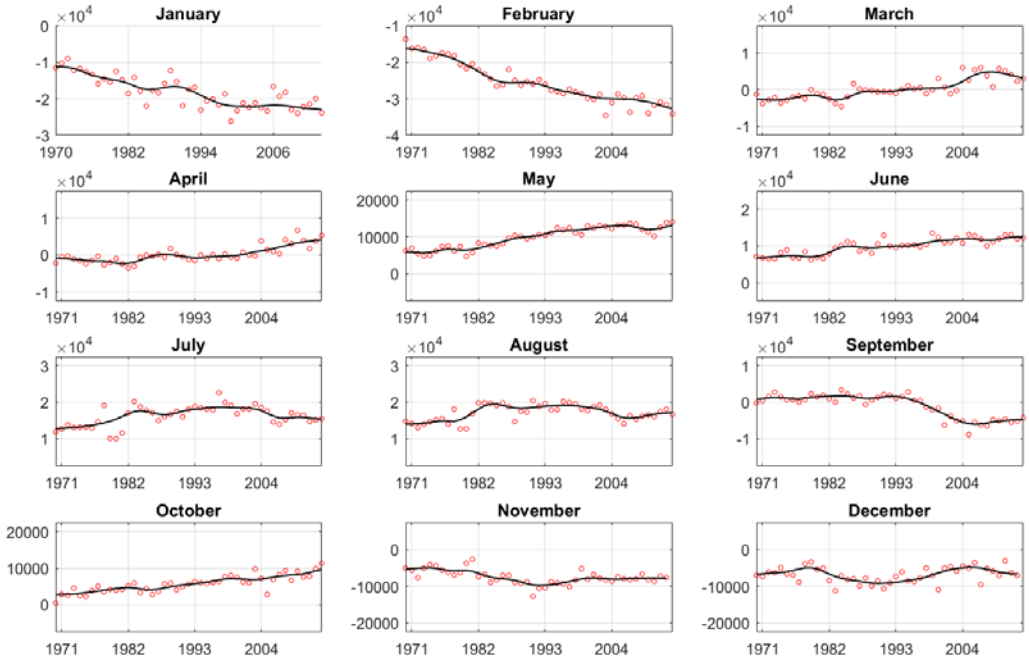
```
--------------------------------------------------------------------------------
 Step 4: Performing the seasonal filtering

 The seasonal factors are rather stable (the graph on the top right shows
 little variation). The decomposition (top left graph) looks reasonable.

 A closer inspection into possible seasonal breaks reveals no major problems
 either. This graph shows the seasonal factors and the SI ratios separately
 for each month. We do see some quantitatively important shifts, but they are
 all slow enough so that the X-11 procedure can deal with it. We do not need
 to specify seasonal breaks in the estimation.
```


miles traveled : dat e2 d12
×10^5
3
2.5
2
1.5
1
0.5
1970 1975 1980 1985 1990 1995 2000 2005 2010


miles traveled : final seasonal factors
×10^4
2
1
0
-1
-2
-3
-4
1 2 3 4 5 6 7 8 9 10 11 12


miles traveled : spr sp1 sp2
75
70
65
60
55
50
1 2 3 4td 5td 6


miles traveled : final irregular component
15000
10000
5000
0
-5000
1 2 3 4 5 6 7 8 9 10 11 12

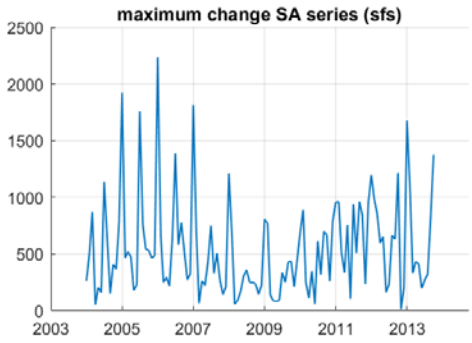

January
February
March
April
May
June
July
August
September
October
November
December
×10^4
1971 1982 1993 2004

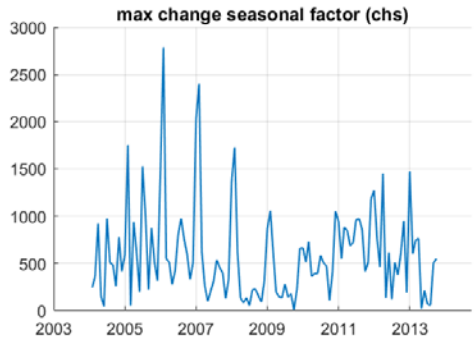
Step 5: Check Stability

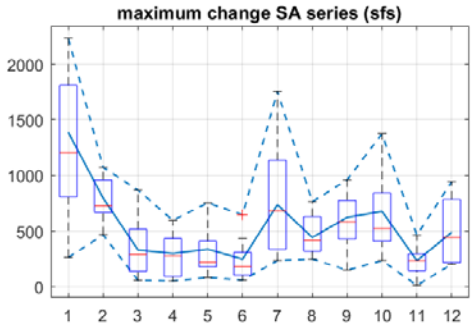
```
disp(sline)
fprintf(' Step 5: Checking stability (this takes a while...)\n\n');

spec5 = makespec(spec4,'SLIDING','HISTORY');
travel5 = x13([travel.dates,travel.data],spec5,'quiet');

% --- sliding span analysis

figure('Position',size4,'Name',[name,': sliding span analysis']);

ax = subplot(2,2,1);
[~,ax] = plot(ax,travel5,'sfs','selection',[0 0 0 0 1]);
title(ax,'\bfmaximum change SA series (sfs)');
ax = subplot(2,2,2);
[~,ax] = plot(ax,travel5,'chs','selection',[0 0 0 0 1]);
title(ax,'\bfmax change seasonal factor (chs)');

ax = subplot(2,2,3);
[~,ax] = plot(ax,travel5,'sfs','selection',[0 0 0 0 1],'span','boxplot');
title(ax,'\bfmaximum change SA series (sfs)');
ax = subplot(2,2,4);
[~,ax] = plot(ax,travel5,'chs','selection',[0 0 0 0 1],'span','boxplot');
title(ax,'\bfmax change seasonal factor (chs)');

report(['CONCLUSION: The sliding span analysis reveals small changes ', ...
    'of the seasonally adjusted series or the seasonal factors. ', ...
    'The maximum revisions are about 2''000, and the level of ', ...
    'the data is between 100''000 and 250''000, so the revisions ', ...
    'amount to about 1%.']);

% --- stability analysis

figure('Position',size4,'Name',[name,': stability analysis']);

ax = subplot(2,2,1);
plot(ax,travel5,'sar')
title(ax,{'\bfmax % change of final vs','concurrent SA series (sar)'});
% % Note: sar = (final./concurrent-1)*100, where
% final = travel4.sae.Final_SA;
% concurrent = travel4.sae.Conc_SA;
% d = travel4.sar.SA_revision-(final./concurrent-1)*100;
% d is equal to zero, except for numerical noise.
ax = subplot(2,2,3);
plot(ax,travel5,'sar','span','boxplot')
title(ax,{'\bfmax % change of final vs','concurrent SA series (sar)'});

ax = subplot(2,2,2);
plot(ax,travel5,'sar','from',datenum(1985,1,1))
title(ax,{'\bf... since 1985'});
ax = subplot(2,2,4);
plot(ax,travel5,'sar','span','boxplot','from',datenum(1985,1,1))
```

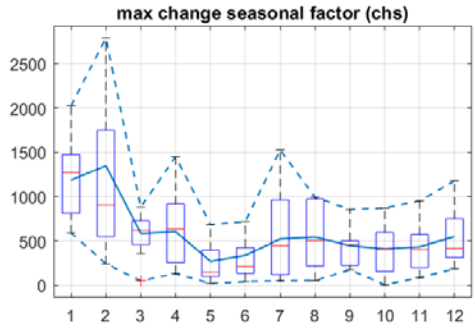
```
title(ax,{'\bf... since 1985'});

report(['CONCLUSION: The historical analysis also reveals small ', ...
    'changes of the seasonally adjusted series, except in the ', ...
    'beginning of the sample in the late 70s, early 80s.']);
```

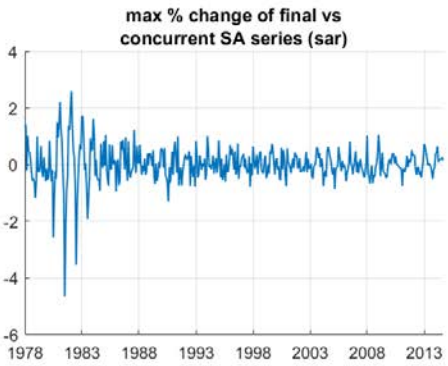
```
--------------------------------------------------------------------------------
 Step 5: Checking stability (this takes a while...)

 CONCLUSION: The sliding span analysis reveals small changes of the seasonally
 adjusted series or the seasonal factors. The maximum revisions are about
 2'000, and the level of the data is between 100'000 and 250'000, so the
 revisions amount to about 1%.

 CONCLUSION: The historical analysis also reveals small changes of the
 seasonally adjusted series, except in the beginning of the sample in the late
 70s, early 80s.
```

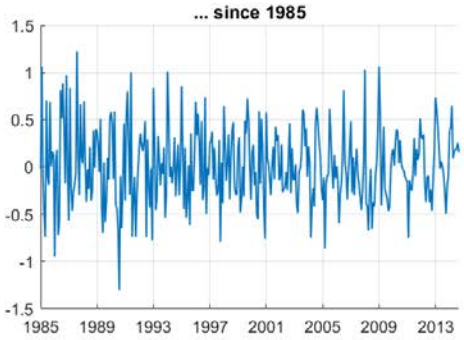


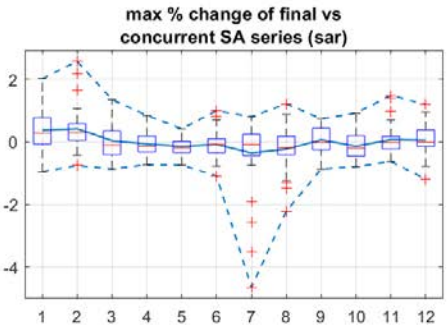


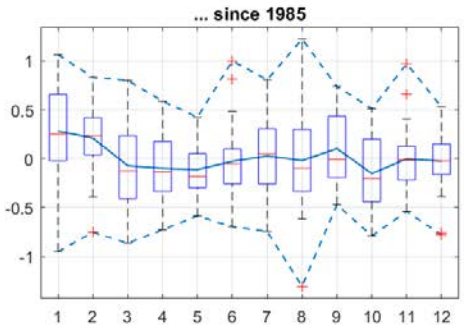












Step 6: Adjust Length Of Filter

```
disp(sline)
fprintf(' Step 6: Adjusting the length of the seasonal filter\n\n');

disp(travel5.table('d9a'));
report(['The X11 procedure selects the length of the filter ', ...
    'according to the global moving seasonality ratio, GMSR. ', ...
    'For a GMSR above 3.5, X11 selects a 3x5 filter, for GMSR below ', ...
    '2.5 it selects a 3x3 filter. Values between 2.5 and 3.5 are in ', ...
    'a grey area, and I don''t know how the filter is selected then.']);

report(['The GMSR for February indicates 3x3 filter for that month. ', ...
    'January, July, and August are in the grey area. However, we can ', ...
    'marginally increase the stability of the filtering by enforcing ', ...
    'a 3x5 filter for all months.']);

spec6 = makespec(spec5,'x11','seasonalma','s3x5');
travel6 = x13([travel.dates,travel.data],spec6, ...
    'graphicsloc',grloc,'quiet');

% --- sliding span and stability analysis

figure('Position',size6,'Name',[name,': sliding span analysis']);

ax = subplot(2,3,1);
```

```
[~,ax] = plot(ax,travel5,travel6,'sfs','selection',[0 0 0 0 1],'comb');
title(ax,'\bfmaximum change SA series (sfs)');

% On my computer, the series I'm looking for is called 'Max___DIFF', but on
% others, strangely, it is called ''Max_0x25_DIFF''. To make this computer-
% independent, I look up the fieldnames.
fn5 = fieldnames(travel5.sfs);
fn6 = fieldnames(travel6.sfs);
ax = subplot(2,3,4);
scatter(ax,travel5.sfs.(fn5{end}),travel6.sfs.(fn6{end}),'.');
hold on; plot(xlim,xlim,'k'); grid on;
xlabel(ax,'sfs spec #5');
ylabel(ax,'sfs spec #6');

ax = subplot(2,3,2);
[~,ax] = plot(ax,travel5,travel6,'chs','selection',[0 0 0 0 1],'comb');
title(ax,'\bfmax change seasonal factor (chs)');

fn5 = fieldnames(travel5.chs);
fn6 = fieldnames(travel6.chs);
ax = subplot(2,3,5);
plot(ax,travel5.chs.(fn5{end}),travel6.chs.(fn6{end}),'.');
hold on; plot(xlim,xlim,'k'); grid on;
xlabel(ax,'chs spec #5');
ylabel(ax,'chs spec #6');

ax = subplot(2,3,3);
[~,ax] = plot(ax,travel5,travel6,'sar','comb');
title(ax,{'\bfmax % change of final vs','concurrent SA series (sar)'});

ax = subplot(2,3,6);
plot(ax,travel5.sar.SA_revision,travel6.sar.SA_revision,'.');
hold on; plot(xlim,xlim,'k'); grid on;
xlabel(ax,'sar spec #5');
ylabel(ax,'sar spec #6');
drawnow;

report(['The difference is minimal, but the largest deviations for ', ...
    'spec #5 are made a bit smaller with spec #6.']);
```

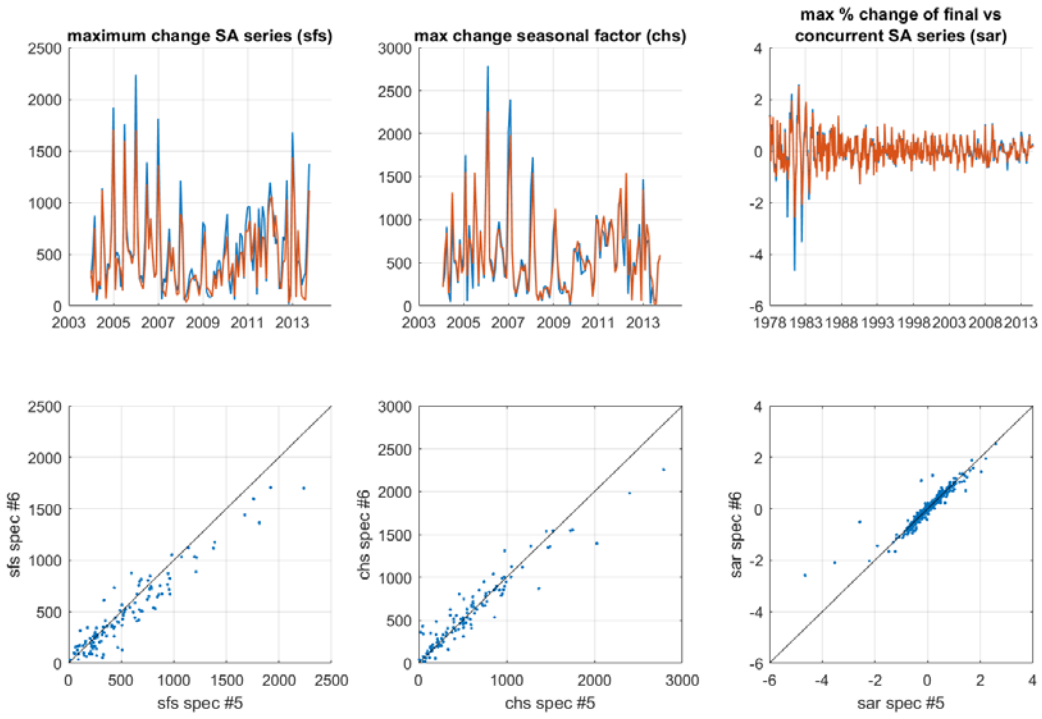
```
--------------------------------------------------------------------------------
 Step 6: Adjusting the length of the seasonal filter

 D 9.A  Moving seasonality ratio
 ------------------------------------------------------------------
              Jan      Feb      Mar      Apr      May      Jun
 ------------------------------------------------------------------
   I       1201.334  945.124 1024.077 1115.102  871.960  929.482
   S        337.512  389.044  273.602  224.123  215.010  187.747
 RATIO        3.559    2.429    3.743    4.975    4.055    4.951

 ------------------------------------------------------------------
              Jul      Aug      Sep      Oct      Nov      Dec
```

```
-------------------------------------------------------------
  I        896.815 1026.743  910.638  983.689  928.668 1030.116
  S        262.301  293.870  255.315  199.126  197.197  278.247
RATIO        3.419    3.494    3.567    4.940    4.709    3.702
```

The X11 procedure selects the length of the filter according to the global moving seasonality ratio, GMSR. For a GMSR above 3.5, X11 selects a 3x5 filter, for GMSR below 2.5 it selects a 3x3 filter. Values between 2.5 and 3.5 are in a grey area, and I don't know how the filter is selected then.

The GMSR for February indicates 3x3 filter for that month. January, July, and August are in the grey area. However, we can marginally increase the stability of the filtering by enforcing a 3x5 filter for all months.

The difference is minimal, but the largest deviations for spec #5 are made a bit smaller with spec #6.



Final Step: specification for production

```
% remove history and sliding spans
spec7 = makespec(spec6, 'history',[],[], 'slidingspans',[],[]);
travel7 = x13([travel.dates,travel.data],spec7,'quiet');

travel7html = x13([travel.dates,travel.data],spec7,'html','quiet');
report('You can view results with web(travel7html.out).')

% report
```

```
disp(travel7.spec);
disp(travel7);

disp(travel7.x2d);
disp(travel7.table('tukey'));

figure('Position',size1,'name','X11')
ax = subplot(3,2,1);
plot(ax,travel7,'dat','e2','d12','comb');
ax = subplot(3,2,3);
plot(ax,travel7,'acf','pcf','comb');
ax = subplot(3,2,5);
plot(ax,travel7,'d13','boxplot','span');
ax = subplot(3,2,2);
plot(ax,travel7,'spr','str','comb');
ax = subplot(3,2,4);
plot(ax,travel7,'sp1','st1','comb');
ax = subplot(3,2,6);
plot(ax,travel7,'sp2','st2','comb');

fh = figure('Position',size6);
seasbreaks(fh,travel7);

report('CONCLUSION: The decomposition appears acceptable.');

% % SEATS cannot deal with four lags (or with missing lags for that
% % matter), so we need to specify a shorter model.
%
% spec8 = makespec(spec7,'TRAMO','SEATS');
% travel8 = x13([travel.dates,travel.data],spec8,'quiet');
%
% figure('Position',size1,'name','SEATS')
% ax = subplot(3,2,1);
% plot(ax,travel8,'dat','s11','s12','comb');
% ax = subplot(3,2,3);
% plot(ax,travel8,'acf','pcf','comb');
% ax = subplot(3,2,5);
% plot(ax,travel8,'s13','boxplot');
% ax = subplot(3,2,2);
% plot(ax,travel8,'spr','str','comb');
% ax = subplot(3,2,4);
% plot(ax,travel8,'s1s','t1s','comb');
% ax = subplot(3,2,6);
% plot(ax,travel8,'s2s','t2s','comb');

disp(dline);
```

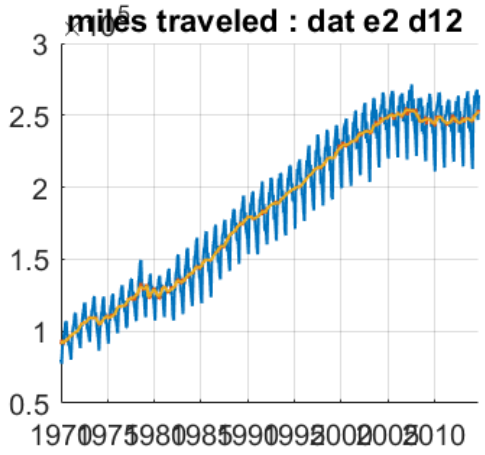
```
 You can view results with web(travel7html.out).

================================================================================
 X-13/X-12 specification object
................................................................................
 - series
```

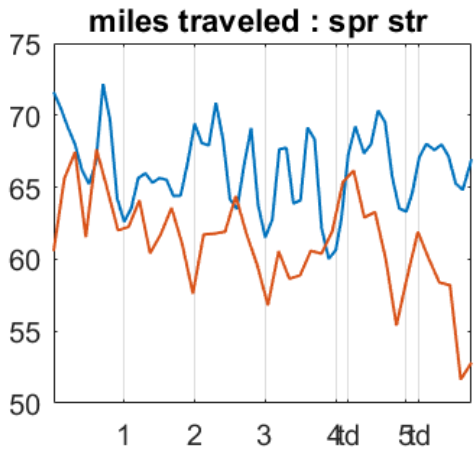
```
   ├─ start : 1970.1
   ├─ modelspan : (1970.1, 2014.10)
   ├─ period : 12
   ├─ file : C:\Users\Yvan\AppData\Local\Temp\X13\milesTraveled.dat
   ├─ precision : 0
   ├─ print : hdr
   ├─ name : milesTraveled
   └─ title : miles traveled
- arima
   └─ model : (0 1 [1 4])(2 1 2)
- check
   ├─ save : (acf ac2 pcf)
   └─ print : (hst nrm)
- estimate
   ├─ save : (est lks mdl ref rsd rts)
   └─ print : (est lks rts)
- outlier
   ├─ print : hdr
   └─ types : none
- regression
   ├─ variables : (AO1993.May AO1995.Jan LS1979.May easter[15] labor[8] td
   |  thank[1])
   └─ save : (ao hol ls td)
- spectrum
   ├─ print : tpk
   └─ save : (sp0 sp1 sp2 spr st0 st1 st2 str)
- transform
   └─ function : none
- x11
   ├─ print : (d8f d9a f2 f3 rsf)
   ├─ save : (d10 d11 d12 d13 d16 d4 d8 e2 e3)
   └─ seasonalma : s3x5

================================================================================
 X-13ARIMA-SEATS object
 Version Number 1.1 Build 39 (x13as.exe)
--------------------------------------------------------------------------------
  Name : miles traveled
  Span : 1970.1 to 2014.10, monthly data
  Data : 538 observations
 Model : (0 1 [1 4])(2 1 2)
--------------------------------------------------------------------------------
 Time Series
 - ao  : regARIMA additive (or point) outlier factors (table A8.AO)
 - dat : unfiltered data
 - d4  : modified SI ratios, D iteration
 - d8  : final unmodified SI ratios (differences)
 - d10 : final seasonal factors
 - d11 : final seasonally adjusted data
 - d12 : final trend-cycle
 - d13 : final irregular component
 - d16 : combined adjustment factors
 - e2  : modified seasonally adjusted series
 - e3  : modified irregular series
```

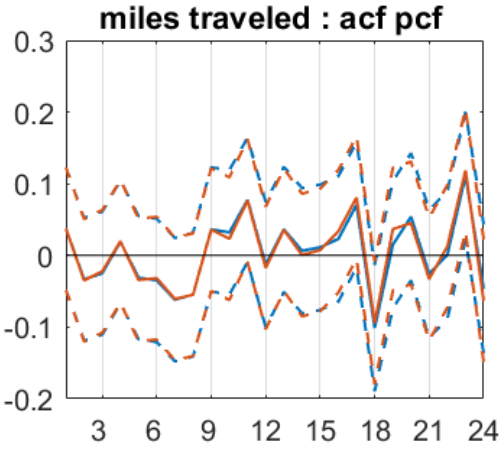
```
 - hol : regARIMA holiday factors (table A7)
 - ls  : regARIMA level change outlier component
 - ref : estimated regression effects (X'beta)
 - rsd : residuals from the estimated model
 - td  : regARIMA trading day component
.....................................................................
 ACF and PACF
 - acf : residual autocorrelations
 - ac2 : squared residual autocorrelations
 - pcf : residual partial autocorrelation
.....................................................................
 Spectra
 - spr : spectrum of the regARIMA model residuals
 - sp0 : spectrum of the first-differenced original series
 - sp1 : spectrum of differenced seasonally adjusted series
 - sp2 : spectrum of modified irregular series
 - str : Tukey spectral estimates of regARIMA model residuals
 - st0 : Tukey spectral estimates of first-differenced original series
 - st1 : Tukey spectral estimates of differenced seasonally adjusted series
 - st2 : Tukey spectral estimates of irregular series
.....................................................................
 Text Items
 - con : console output
 - err : program error file
 - est : regression and ARMA parameter estimates, with standard errors
 - lks : log-likelihood at final parameter estimates and, if exact = arma is
         used (default option), corresponding model selection criteria (AIC,
         AICC, Hannan-Quinn, BIC)
 - log : program log file
 - mdl : regression and arima specs corresponding to the model, with the
         estimation results used to specify initial values for the ARMA
         parameters
 - out : program output file
 - rts : roots of the AR and MA operators
 - spc : specification file
 - udg : diagnostics summary file
 - x2d : seasonal adjustment diagnostics
.....................................................................
 Tables
    heading : U. S. Department of Commerce, U. S. Census Bureau
       eval : MODEL ESTIMATION/EVALUATION
 regression : Estimation converged in   21 ARMA iterations,  226 function eval
 diagnostic : DIAGNOSTIC CHECKING
        d8a :   D 8.A  F-tests for seasonality
        d9a :  D 9.A  Moving seasonality ratio
  residseas : Test for the presence of residual seasonality.
         f2 :   F 2. Summary Measures
        f2a :    F 2.A: Average differences without regard to sign over the
        f2b :    F 2.B: Relative contributions to the variance of the    diffe
        f2c :    F 2.C: Average differences with regard to sign and standard
        f2d :    F 2.D: Average duration of run        CI      I       C
        f2e :    F 2.E: I/C Ratio for months span
        f2f :    F 2.F: Relative contribution of the components to the station
        f2g :    F 2.G: The autocorrelation of the irregulars for spans 1 to 1
```

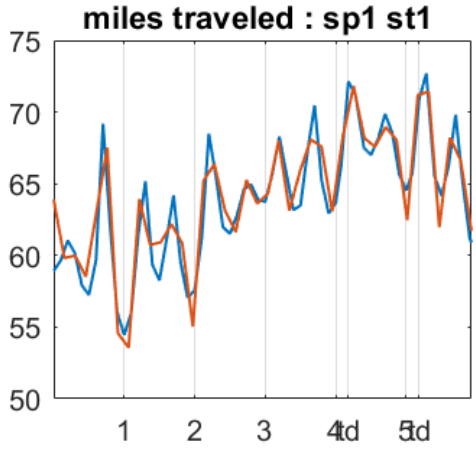
```
       f2h :    F 2.H: The final I/C Ratio from Table D12:        2.07
       f2i :    F 2.I:                                                Sta
        f3 :   F 3. Monitoring and Quality Assessment Statistics
     tukey : Peak probabilities for Tukey spectrum estimator
.....................................................................
 NOTE: Use obj.table('name') to see content of a table, where 'name' can be
 abbreviated.
.....................................................................
 Time of run: 04-Oct-2018 23:44:11 (3.1 sec)
=====================================================================

 Series name: milesTraveled
 Span used:  1st month,1970 to 10th month,2014

 X-11 Seasonal Adjustment

 Seasonal filter length:
     3x5   3x5   3x5   3x5   3x5   3x5   3x5   3x5   3x5   3x5   3x5   3x5
 Trend filter length: default
 X-11 Extreme adjustment:  Standard Error
 Type of Adjustment: additive seasonal adjustment

 Diagnostics for direct seasonally adjusted series

 F-test for Stable Seasonality (D8):            846.211
 F-test for Moving Seasonality (D8):              7.145
 Identifiable Seasonality present

 Relative Contribution of the Irregular to the Variance

              I       C       S       P       TD    TOTAL
 F2.F    :    0.35   37.33   63.69    4.56    0.28  106.21

 I/C RATIO:         2.07  I/S RATIO:         3.85

 M1:   0.041  M2:   0.037  M3:   0.537  M4:   0.132
 M5:   0.480  M6:   0.059  M7:   0.130  M8:   0.193
 M9:   0.108  M10:  0.163  M11:  0.147
 Q =  0.19
 Q(without M2) =  0.21

 Visually distinct seasonal spectral peaks were not found.

 Visually distinct trading day spectral peaks were not found.

Peak probabilities for Tukey spectrum estimator
  Spectrum estimated from 2006.Nov to 2014.Oct.

                                      S1       S2       S3       S4       S5       S6
TD
                                    ------   ------   ------   ------   ------   ------   ---
---
  Model Residuals                    0.419    0.134    0.168    0.331    0.409    0.662
0.783
```

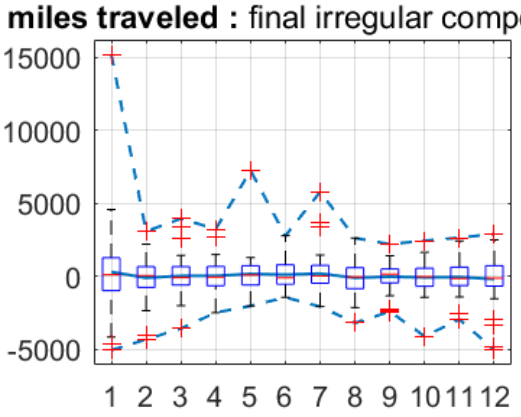
```
  Prior Adjusted Series (Table B1)       0.960*   0.999**  0.998**  0.988*   1.000**  0.822
0.021
  Seasonally adjusted series (E2)        0.014    0.009    0.284    0.077    0.018    0.239
0.924*
  Modified Irregular (E3)                0.170    0.015    0.278    0.080    0.021    0.233
0.941*
  ----------
     ** - Peak Probability > 0.99,
      * - 0.90 < Peak Probability < 0.99

 CONCLUSION: The decomposition appears acceptable.

================================================================================
```

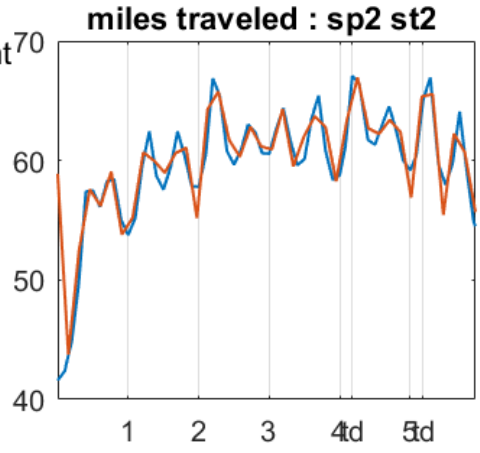

miles traveled : dat e2 d12
×10^5
3
2.5
2
1.5
1
0.5
1970
1975
1980
1985
1990
1995
2000
2005
2010

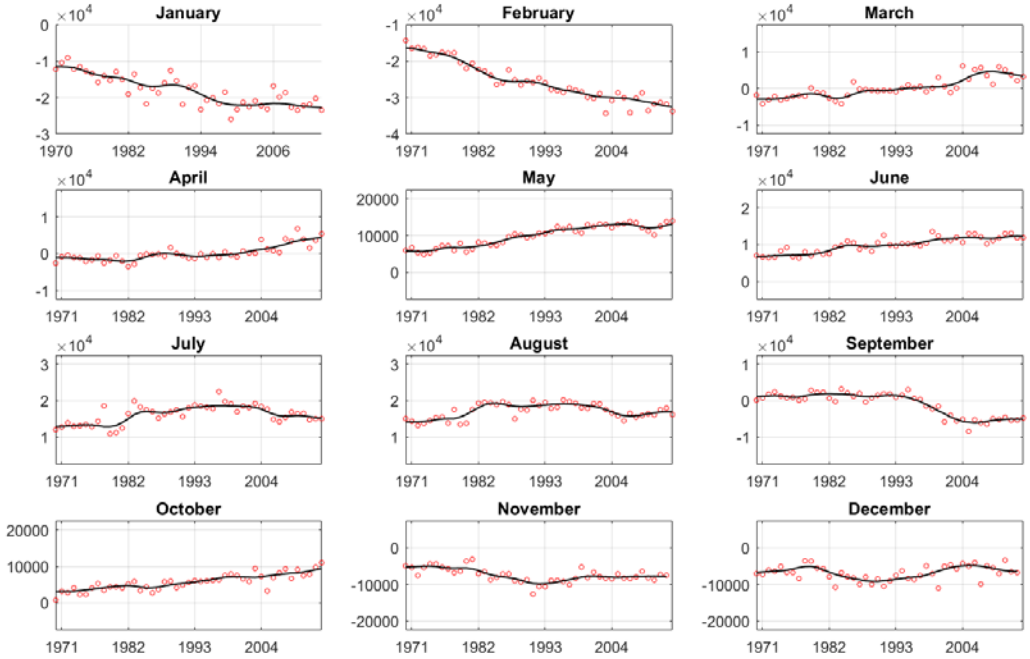

miles traveled : spr str
75
70
65
60
55
50
1
2
3
4td
5td


miles traveled : acf pcf
0.3
0.2
0.1
0
-0.1
-0.2
3
6
9
12
15
18
21
24


miles traveled : sp1 st1
75
70
65
60
55
50
1
2
3
4td
5td


miles traveled : final irregular component
15000
10000
5000
0
-5000
1
2
3
4
5
6
7
8
9
10
11
12


miles traveled : sp2 st2
70
60
50
40
1
2
3
4td
5td



Published with MATLAB® R2017a

DEMO for X13 Toolbox: Composite Run

```
% turn warnings temporarily off
orig_warning_state = warning('off','all');
```

PRELIMINARIES

```
% get correct path
p = fileparts(mfilename('fullpath'));   % directory of this m-file
% if the section is run with Shift-Ctrl-Enter ...
if isempty(p); p = [cd,'\']; end
% location of graphics files for use with X-13-Graph program
grloc = fullfile(p,'graphics\');

% size for figures with subplots
scSize = get(groot,'ScreenSize');
scWidth = scSize(3); scHeight = scSize(4);
sizeFig = @(h,v) round([0.04*scWidth, scHeight*(1-0.08-v/100)-50, ...
    h/100*scWidth, v/100*scHeight]);

size1 = sizeFig(40,80);
size2 = sizeFig(80,45);
size3 = sizeFig(95,45);
size4 = sizeFig(70,70);
size6 = sizeFig(75,68);
size8 = sizeFig(95,65);
size9 = sizeFig(95,90);

% line width
lwidth = 78;

% single and double line
sline = repmat('-',1,lwidth+1);
dline = repmat('=',1,lwidth+1);

% display with wrapped lines and leading space
```

```
report = @(s) disp(WrapLines(s,lwidth,' '));

% write heading
clc; disp(dline);
report(['DEMONSTRATION OF X-13 TOOLBOX FOR MATLAB : ', ...
    'composite run']);
report(['This script was developed with MATLAB Version ', ...
    '8.3.0.532 (R2014a)']);
disp(sline)
```

```
================================================================================
 DEMONSTRATION OF X-13 TOOLBOX FOR MATLAB : composite run

 This script was developed with MATLAB Version 8.3.0.532 (R2014a)

--------------------------------------------------------------------------------
```

LOADING DATA

```
load(fullfile(p,'gdp'));
report(['Data from Eurostat: quarterly GDP for several countries ', ...
    'http://ec.europa.eu/eurostat/web/national-accounts/data/database']);
name = 'Nominal GDP European Countries';

ctry  = gdp{1};
dates = gdp{2};
data  = gdp{3};

remove = any(isnan(data),2);
data(remove,:) = [];
dates(remove)  = [];

disp(sline)
```

```
 Data from Eurostat: quarterly GDP for several countries
 http://ec.europa.eu/eurostat/web/national-accounts/data/database

--------------------------------------------------------------------------------
```

COMPOSITE RUN

```
report(['We do a composite run of the GDPs of some European ', ...
    'countries. We seasonally adjust the aggregate GDP (direct ', ...
    'seasonal adjustment) and we seasonally adjust the individual ', ...
    'components and aggregate them afterwards (indirect seasonal ', ...
    'adjustment).']);
n = numel(ctry);

% specifications for the individual series
spec = cell(1,n);
for c = 1:n
    spec{c} = makespec('MULT','PICKBEST','X11','LS','ACF', ...
```

```
        'x11','appendfcst','no', 'series','comptype','add', ...
        'series','name',ctry{c});
end

compspec = makespec('MULT','PICKBEST','X11','LS','ACF', ...
    'x11','appendfcst','no', ...
    'outlier','method','addone', ...
    'composite','name','Aggregate', ...
    'composite','save','(cms itn isa iir iaf)');

% run x13 and show result
xgdp = x13([dates,data],spec,compspec,'graphicsloc',grloc);
disp(xgdp);
```

```
 We do a composite run of the GDPs of some European countries. We seasonally
 adjust the aggregate GDP (direct seasonal adjustment) and we seasonally
 adjust the individual components and aggregate them afterwards (indirect
 seasonal adjustment).

================================================================================
 X-13-ARIMA-SEATS composite object
 Version Number 1.1 Build 19
................................................................................
 List of series:
-> Aggregate
 - Austria
 - Belgium
 - Bulgaria
 - Croatia
 - Cyprus
 - Czech Republic   [.CzechRepublic]
 - Denmark
 - Estonia
 - Finland
 - France
 - Germany
 - Greece
 - Hungary
 - Iceland
 - Ireland
 - Italy
 - Latvia
 - Lithuania
 - Luxembourg
 - Malta
 - Netherlands
 - Norway
 - Poland
 - Portugal
 - Romania
 - Slovakia
 - Slovenia
 - Spain
```

```
 - Sweden
 - Switzerland
 - Turkey
 - United Kingdom   [.UnitedKingdom]
.....................................................................................
 Time of run: 25-Jul-2015 00:02:16 (33.4 sec)
=====================================================================================
```

REPORT SOME RESULTS

```
allseries = xgdp.listofseries;

fprintf('\n --- KEY STATISTICS ------------------------------------------------------------\n');
for c = 1:numel(allseries)
    strOUT   = xgdp.(allseries{c}).x2d;
    strOUT   = strrep(strOUT,char(10),[char(32),char(10)]);
    linesOUT = strsplit(strOUT,char(10));
    strD8A   = xgdp.(allseries{c}).table('d8a');
    linesD8A = strsplit(strD8A,char(10));
    fprintf('\n *** %s ***\n Model: %s\n', ...
        upper(xgdp.(allseries{c}).name), ...
        xgdp.(allseries{c}).arima)
    fprintf('\n %s\n\n',strtrim(lower(linesD8A{end-1})));
    if strcmp(xgdp.(allseries{c}).name,xgdp.compositeseries)
        first = -17;
    else
        first = -5;
    end
    for l = first:1:0
        disp(linesOUT{end+l});
    end
end
fprintf('\n -------------------------------------------------------------------------------\n\n');

report(['Most M-statistics pass, with some exceptions. The ', ...
    'M4-statistic fails (>1) for several countries, but this is the ', ...
    'least reliable M-statistic, so we ignore it. The M1-statistic ', ...
    'marginally fails for the United Kingdom, as does M8 for several ', ...
    'countries. We ignore this as well. You are welcome to look for ', ...
    'improvements']);
report(['Furthermore, PICKMDL did not find a good model for Finland. ', ...
    'You might want to experiment with that (you can do so by ', ...
    'analyzing the Finnish data separately; you don''t need to embed ', ...
    'it in a composite while searching for a good model for the ', ...
    'Finnsh series). Maybe TRAMO finds a usable model? For this ', ...
    'demo, we ignore this problem.']);
fprintf(' -------------------------------------------------------------------------------\n\n');
```

```
 --- KEY STATISTICS ------------------------------------------------------------

 *** AGGREGATE ***
```

Model: (2 1 0)(0 1 2)

identifiable seasonality present

M1: 0.071 M2: 0.030 M3: 0.000 M4: 0.165
M5: 0.200 M6: 0.743 M7: 0.079 M8: 0.238
M9: 0.134 M10: 0.314 M11: 0.245
Q = 0.18
Q(without M2) = 0.20

Comparison between Smoothness Measures for Direct and Indirect Adjustments

	(Dir - Ind) Percentage Change	
	Entire srs.	Last 3 yrs.
R1 - MSE	4.131	1.204
R1 - RMSE	2.087	0.604
R2 - MSE	85.718	11.793
R2 - RMSE	62.209	6.081

Positive percentage changes indicate that the indirect seasonally adjusted
composite is smoother than the direct seasonally adjusted composite.

*** AUSTRIA ***
Model: (2 0 0)(0 1 2)

identifiable seasonality present

M1: 0.186 M2: 0.089 M3: 0.039 M4: 0.785
M5: 0.200 M6: 0.128 M7: 0.176 M8: 0.324
M9: 0.258 M10: 0.319 M11: 0.319
Q = 0.23
Q(without M2) = 0.24

*** BELGIUM ***
Model: (1 0 0)(0 1 1)

identifiable seasonality present

M1: 0.011 M2: 0.016 M3: 0.000 M4: 0.909
M5: 0.200 M6: 0.173 M7: 0.058 M8: 0.130
M9: 0.083 M10: 0.182 M11: 0.182
Q = 0.15
Q(without M2) = 0.17

*** BULGARIA ***
Model: (0 0 2)(2 1 0)

identifiable seasonality present

M1: 0.041 M2: 0.021 M3: 0.003 M4: 0.909
M5: 0.200 M6: 0.391 M7: 0.073 M8: 0.193
M9: 0.169 M10: 0.221 M11: 0.221
Q = 0.20
Q(without M2) = 0.22

```
*** CROATIA ***
Model: (0 1 1)(1 1 0)

identifiable seasonality present

M1:   0.020  M2:   0.006  M3:   0.000  M4:   0.909
M5:   0.200  M6:   0.781  M7:   0.095  M8:   0.407
M9:   0.105  M10:  0.632  M11:  0.597
Q =  0.22
Q(without M2) =  0.25

*** CYPRUS ***
Model: (0 1 2)(0 1 0)

identifiable seasonality present

M1:   0.046  M2:   0.006  M3:   0.000  M4:   0.909
M5:   0.200  M6:   0.611  M7:   0.109  M8:   0.357
M9:   0.179  M10:  0.262  M11:  0.239
Q =  0.20
Q(without M2) =  0.22

*** CZECH REPUBLIC ***
Model: (0 0 2)(0 1 0)

identifiable seasonality present

M1:   0.058  M2:   0.012  M3:   0.000  M4:   0.165
M5:   0.200  M6:   0.175  M7:   0.137  M8:   0.324
M9:   0.096  M10:  0.260  M11:  0.205
Q =  0.13
Q(without M2) =  0.15

*** DENMARK ***
Model: (1 0 0)(0 1 2)

identifiable seasonality present

M1:   0.105  M2:   0.081  M3:   0.138  M4:   0.537
M5:   0.200  M6:   0.321  M7:   0.159  M8:   0.399
M9:   0.387  M10:  0.431  M11:  0.431
Q =  0.25
Q(without M2) =  0.27

*** ESTONIA ***
Model: (2 1 0)(0 1 1)

identifiable seasonality present

M1:   0.307  M2:   0.041  M3:   0.040  M4:   1.404
M5:   0.200  M6:   0.259  M7:   0.211  M8:   0.656
M9:   0.281  M10:  0.423  M11:  0.311
Q =  0.33
```

```
Q(without M2) =  0.37

*** FINLAND ***
Model: No good model found, using default: (0 1 1)(0 1 1)

identifiable seasonality present

M1:   0.166  M2:   0.097  M3:   0.317  M4:   1.156
M5:   0.200  M6:   0.459  M7:   0.115  M8:   0.234
M9:   0.161  M10:  0.252  M11:  0.245
Q =  0.29
Q(without M2) =  0.31

*** FRANCE ***
Model: (1 0 0)(2 1 0)

identifiable seasonality present

M1:   0.027  M2:   0.015  M3:   0.000  M4:   0.413
M5:   0.200  M6:   0.115  M7:   0.081  M8:   0.182
M9:   0.147  M10:  0.151  M11:  0.147
Q =  0.12
Q(without M2) =  0.13

*** GERMANY ***
Model: (2 1 0)(2 1 0)

identifiable seasonality present

M1:   0.226  M2:   0.128  M3:   0.098  M4:   0.413
M5:   0.200  M6:   0.037  M7:   0.163  M8:   0.336
M9:   0.300  M10:  0.226  M11:  0.181
Q =  0.20
Q(without M2) =  0.20

*** GREECE ***
Model: (1 1 0)(1 1 0)

identifiable seasonality present

M1:   0.090  M2:   0.013  M3:   0.000  M4:   0.909
M5:   0.200  M6:   0.583  M7:   0.170  M8:   0.366
M9:   0.329  M10:  0.604  M11:  0.604
Q =  0.29
Q(without M2) =  0.33

*** HUNGARY ***
Model: (1 0 0)(2 1 0)

identifiable seasonality present

M1:   0.111  M2:   0.027  M3:   0.083  M4:   0.909
M5:   0.200  M6:   0.093  M7:   0.105  M8:   0.295
M9:   0.220  M10:  0.471  M11:  0.463
```

```
Q =  0.22
Q(without M2) =  0.24

*** ICELAND ***
Model: (1 1 0)(2 1 0)

identifiable seasonality present

M1:   0.757  M2:   0.037  M3:   0.000  M4:   1.032
M5:   0.200  M6:   0.561  M7:   0.550  M8:   2.079
M9:   0.456  M10:  1.696  M11:  1.696
Q =  0.66
Q(without M2) =  0.75

*** IRELAND ***
Model: (0 1 1)(0 1 2)

identifiable seasonality present

M1:   1.516  M2:   0.058  M3:   0.452  M4:   0.909
M5:   0.351  M6:   0.251  M7:   0.757  M8:   1.458
M9:   0.686  M10:  1.771  M11:  1.553
Q =  0.76
Q(without M2) =  0.85

*** ITALY ***
Model: (0 0 2)(0 1 1)

identifiable seasonality present

M1:   0.021  M2:   0.013  M3:   0.000  M4:   1.032
M5:   0.200  M6:   0.745  M7:   0.133  M8:   0.386
M9:   0.270  M10:  0.563  M11:  0.563
Q =  0.25
Q(without M2) =  0.28

*** LATVIA ***
Model: (0 1 2)(0 1 2)

identifiable seasonality present

M1:   0.041  M2:   0.009  M3:   0.000  M4:   0.413
M5:   0.200  M6:   0.760  M7:   0.147  M8:   0.466
M9:   0.239  M10:  0.481  M11:  0.481
Q =  0.19
Q(without M2) =  0.22

*** LITHUANIA ***
Model: (1 0 0)(0 1 2)

identifiable seasonality present

M1:   0.116  M2:   0.044  M3:   0.000  M4:   0.661
M5:   0.200  M6:   0.687  M7:   0.151  M8:   0.683
```

```
M9:   0.381  M10:  0.625  M11:  0.355
Q =  0.26
Q(without M2) =  0.29

*** LUXEMBOURG ***
Model: (0 1 0)(2 1 0)

identifiable seasonality present

M1:   0.230  M2:   0.086  M3:   0.001  M4:   0.289
M5:   0.200  M6:   0.078  M7:   0.217  M8:   0.536
M9:   0.334  M10:  0.796  M11:  0.796
Q =  0.25
Q(without M2) =  0.27

*** MALTA ***
Model: (0 1 1)(0 1 1)

identifiable seasonality present

M1:   0.237  M2:   0.159  M3:   0.421  M4:   0.909
M5:   0.364  M6:   0.333  M7:   0.189  M8:   0.249
M9:   0.196  M10:  0.309  M11:  0.278
Q =  0.32
Q(without M2) =  0.34

*** NETHERLANDS ***
Model: (2 0 0)(2 1 0)

identifiable seasonality present

M1:   0.013  M2:   0.009  M3:   0.000  M4:   0.413
M5:   0.200  M6:   0.414  M7:   0.099  M8:   0.321
M9:   0.212  M10:  0.235  M11:  0.062
Q =  0.14
Q(without M2) =  0.16

*** NORWAY ***
Model: (1 0 0)(1 1 0)

identifiable seasonality present

M1:   0.366  M2:   0.610  M3:   0.000  M4:   0.785
M5:   0.200  M6:   0.197  M7:   0.340  M8:   1.050
M9:   0.516  M10:  0.772  M11:  0.311
Q =  0.45
Q(without M2) =  0.42

*** POLAND ***
Model: (1 0 0)(2 1 0)

identifiable seasonality present

M1:   0.308  M2:   0.331  M3:   0.086  M4:   1.156
```

```
M5:   0.200  M6:   0.742  M7:   0.137  M8:   0.237
M9:   0.062  M10:  0.258  M11:  0.214
Q =  0.33
Q(without M2) =  0.33

*** PORTUGAL ***
Model: (0 0 1)(0 1 1)

identifiable seasonality present

M1:   0.126  M2:   0.048  M3:   0.252  M4:   1.156
M5:   0.200  M6:   0.446  M7:   0.105  M8:   0.377
M9:   0.180  M10:  0.750  M11:  0.750
Q =  0.32
Q(without M2) =  0.35

*** ROMANIA ***
Model: (0 1 0)(0 1 0)

identifiable seasonality present

M1:   0.044  M2:   0.016  M3:   0.027  M4:   0.661
M5:   0.200  M6:   0.286  M7:   0.048  M8:   0.135
M9:   0.018  M10:  0.201  M11:  0.192
Q =  0.15
Q(without M2) =  0.16

*** SLOVAKIA ***
Model: (2 0 0)(0 1 2)

identifiable seasonality present

M1:   0.097  M2:   0.014  M3:   0.000  M4:   0.165
M5:   0.200  M6:   0.585  M7:   0.165  M8:   0.654
M9:   0.360  M10:  0.383  M11:  0.336
Q =  0.20
Q(without M2) =  0.22

*** SLOVENIA ***
Model: (1 0 0)(2 1 0)

identifiable seasonality present

M1:   0.043  M2:   0.008  M3:   0.000  M4:   0.785
M5:   0.200  M6:   0.626  M7:   0.147  M8:   0.395
M9:   0.334  M10:  0.174  M11:  0.130
Q =  0.20
Q(without M2) =  0.23

*** SPAIN ***
Model: (2 1 0)(2 1 0)

identifiable seasonality present
```

```
M1:   0.019  M2:   0.004  M3:   0.000  M4:   0.785
M5:   0.200  M6:   0.430  M7:   0.088  M8:   0.218
M9:   0.104  M10:  0.230  M11:  0.207
Q =  0.19
Q(without M2) =  0.21

*** SWEDEN ***
Model: (2 0 0)(0 1 1)

identifiable seasonality present

M1:   0.050  M2:   0.099  M3:   0.000  M4:   1.032
M5:   0.200  M6:   0.144  M7:   0.192  M8:   0.312
M9:   0.275  M10:  0.271  M11:  0.268
Q =  0.23
Q(without M2) =  0.25

*** SWITZERLAND ***
Model: (0 1 2)(0 1 0)

identifiable seasonality present

M1:   0.480  M2:   0.014  M3:   0.000  M4:   1.156
M5:   0.200  M6:   0.327  M7:   0.885  M8:   1.529
M9:   0.523  M10:  1.999  M11:  1.892
Q =  0.66
Q(without M2) =  0.73

*** TURKEY ***
Model: (2 0 0)(1 1 0)

identifiable seasonality present

M1:   0.536  M2:   0.179  M3:   0.177  M4:   0.785
M5:   0.200  M6:   0.447  M7:   0.293  M8:   0.612
M9:   0.334  M10:  0.863  M11:  0.863
Q =  0.41
Q(without M2) =  0.44

*** UNITED KINGDOM ***
Model: (0 1 0)(0 1 1)

identifiable seasonality present

M1:   1.209  M2:   0.080  M3:   0.137  M4:   0.413
M5:   0.200  M6:   0.443  M7:   0.585  M8:   1.178
M9:   0.319  M10:  1.421  M11:  1.204
Q =  0.56
Q(without M2) =  0.62

------------------------------------------------------------------------------

Most M-statistics pass, with some exceptions. The M4-statistic fails (>1) for
several countries, but this is the least reliable M-statistic, so we ignore
```

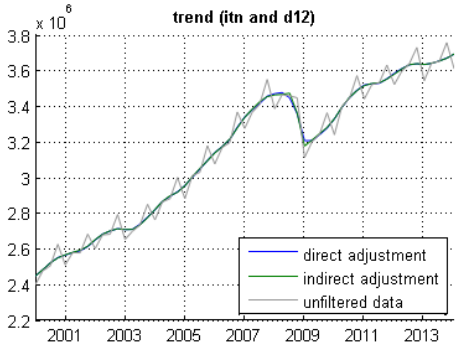
it. The M1-statistic marginally fails for the United Kingdom, as does M8 for several countries. We ignore this as well. You are welcome to look for improvements

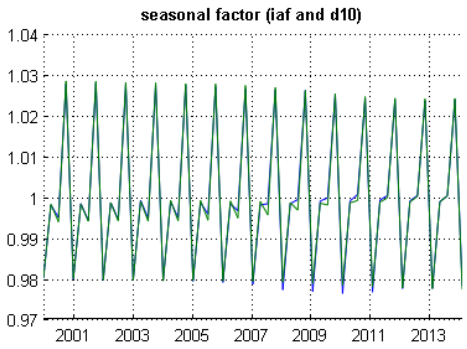
Furthermore, PICKMDL did not find a good model for Finland. You might want to experiment with that (you can do so by analyzing the Finnish data separately; you don't need to embed it in a composite while searching for a good model for the Finnsh series). Maybe TRAMO finds a usable model? For this demo, we ignore this problem.

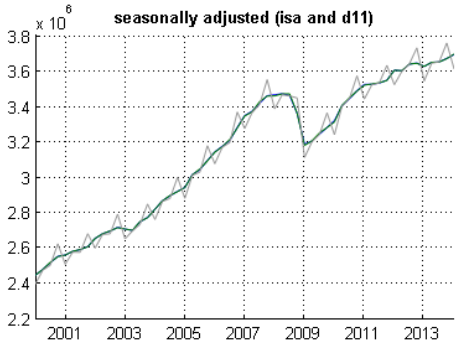
COMPARE DIRECT WITH INDIRECT ADJUSTMENTS

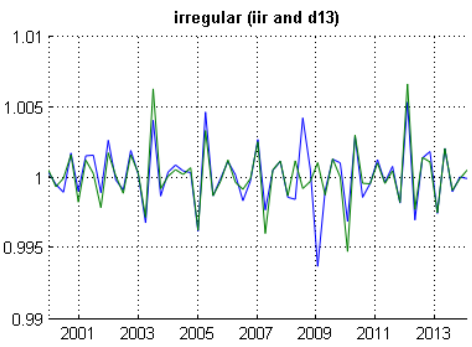
```
figure('Position',size4, ...
    'Name',[name,': comparing direct with indirect adjustment']);

aggr = xgdp.compositeseries;    % name of composite series

ax = subplot(2,2,2);
plot(ax,xgdp.(aggr),'iaf','d10','combined');
title('\bfseasonal factor (iaf and d10)');
%
ax = subplot(2,2,3);
plot(ax,xgdp.(aggr),'isa','d11','combined');
plot(ax,xgdp.(aggr),'cms','combined','options',{'Color',[0.6,0.6,0.6]});
title('\bfseasonally adjusted (isa and d11)');
%
ax = subplot(2,2,1);
plot(ax,xgdp.(aggr),'itn','d12','combined');
plot(ax,xgdp.(aggr),'cms','combined','options',{'Color',[0.6,0.6,0.6]});
legend(ax,'direct adjustment','indirect adjustment','unfiltered data');
legend(ax,'Location','SouthEast');
title('\bftrend (itn and d12)');
%
ax = subplot(2,2,4);
plot(ax,xgdp.(aggr),'iir','d13','combined');
title('\bfirregular (iir and d13)');

report(['CONCLUSION: The differences between direct and indirect ', ...
    'adjustments are small. The irregular components (iir and d13) ', ...
    'look a bit different, but it is not obvious which one is superior.']);
```

CONCLUSION: The differences between direct and indirect adjustments are small. The irregular components (iir and d13) look a bit different, but it is not obvious which one is superior.







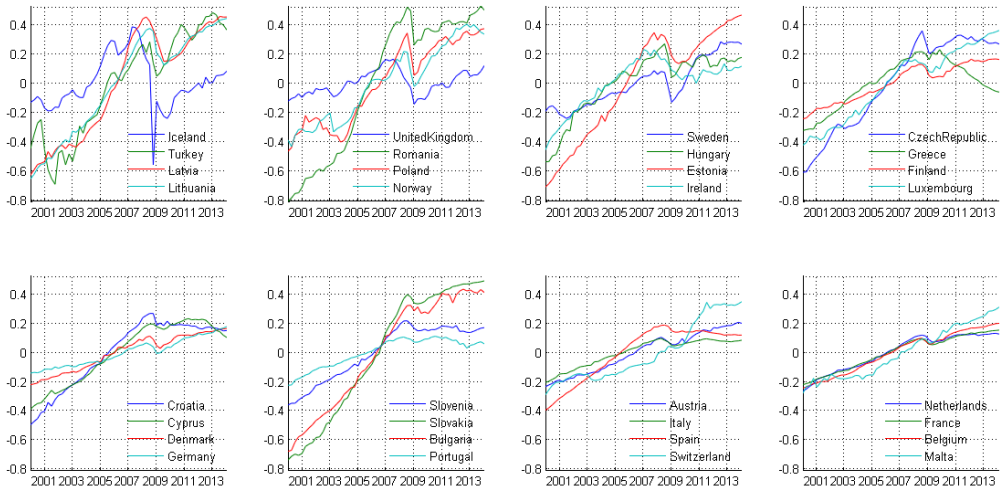


PLOT NORMALIZED LOG GDP FOR ALL COUNTRIES

```
% get list of countries
prop = xgdp.listofseries;
remove = ismember(prop,aggr);
prop(remove) = [];
% sort according to maximum negative relative yoy change
s = nan(1,numel(prop));
for c = 1:numel(prop)
    loggdp = log(xgdp.(prop{c}).d11.d11);
    s(c) = min(loggdp(5:end)-loggdp(1:end-4));
end
[~,ord] = sort(s);
prop = prop(ord);
% plot seasonally adjusted series
% use log scale and normalize means for better comparison
fh = figure('Position',size8);
nax = 8;
n = ceil(numel(prop)/nax);
yl = [0,0];
colorOrder = get(gcf,'DefaultAxesColorOrder');
nColors    = size(colorOrder,1);
for f = 1:nax
    ax(f) = subplot(2,4,f);
    leg = cell(0);
    colorRow   = 0;
```

```
    for c = (f-1)*n+1:min(numel(prop),f*n)
        col = colorOrder(colorRow + 1,:);
        colorRow = mod(colorRow + 1, nColors);
        plot(ax(f),xgdp.(prop{c}),'d11','logscale','meannorm','comb', ...
            'options',{'Color',col});
        hold(ax(f),'all');
        leg{end+1} = prop{c};
    end
    legend(ax(f),leg{:});
    legend(ax(f),'Location','SouthEast');
    legend(ax(f),'boxoff');
    title(ax(f),'');
    axis(ax(f),'tight');
    ylnew = ylim;
    yl(1) = min(yl(1),ylnew(1));
    yl(2) = max(yl(2),ylnew(2));
end
for f = 1:nax
    ylim(ax(f),[yl(1),yl(2)]);
end

report(['CONCLUSION: The normalized log seasonally adjusted levels ', ...
    'give a clear view of how the financial / govt debt crisis has ', ...
    'affected different countries. Some were hit brutally but have ', ...
    'recovered quickly (e.g. Sweden). Others are back on their ', ...
    'pre-crisis growth rates, but their levels seem to have shifted ', ...
    'down permanently (UK, Iceland). Still others have essentially ', ...
    'stalled since the crisis (e.g. Spain). Greece has even ', ...
    'developped a negative (!) trend.']);
```

```
CONCLUSION: The normalized log seasonally adjusted levels give a clear view
of how the financial / govt debt crisis has affected different countries.
Some were hit brutally but have recovered quickly (e.g. Sweden). Others are
back on their pre-crisis growth rates, but their levels seem to have shifted
down permanently (UK, Iceland). Still others have essentially stalled since
the crisis (e.g. Spain). Greece has even developped a negative (!) trend.
```

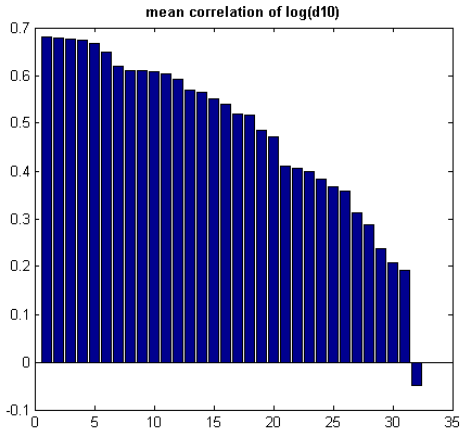


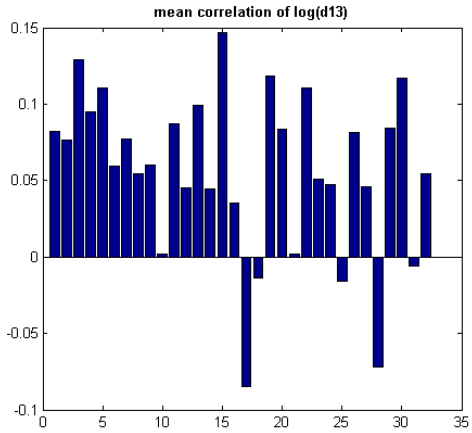
STUDY CORRELATIONS OF HIGH FREQUENCY COMPONENTS

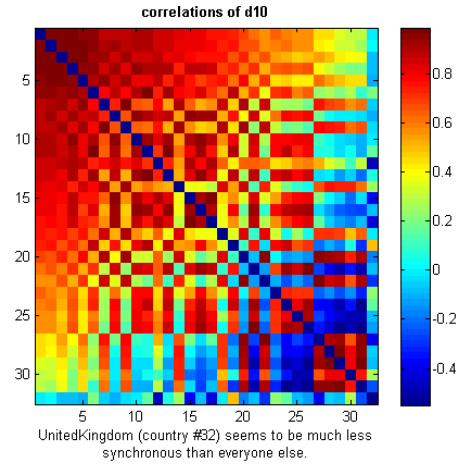
```
% extract variables from x13series objects and place them into arrays
prop = xgdp.listofseries;
remove = ismember(prop,aggr);
prop(remove) = [];
d10 = nan(numel(dates),numel(prop));
d13 = nan(numel(dates),numel(prop));
for c = 1:numel(prop)
   d10(:,c) = log(xgdp.(prop{c}).d10.d10);
   d13(:,c) = log(xgdp.(prop{c}).d13.d13);
end
% mean correlations
figure('Position',size2,'Name',[name,': mean correlations']);
ax = subplot(1,2,1);
meancorr = mean(corr(d10)-diag(ones(1,numel(prop))));
[~,ord] = sort(meancorr,'descend');
bar(ax,meancorr(ord));
title(ax,'\bfmean correlation of log(d10)');
ax = subplot(1,2,2);
meancorr = mean(corr(d13)-diag(ones(1,numel(prop))));
bar(ax,meancorr(ord));
title(ax,'\bfmean correlation of log(d13)');
% study correlations of seasonal and irregular components
figure('Position',size2, ...
    'Name',[name,': pairwise correlations']);
subplot(1,2,1);
imagesc(corr(d10(:,ord)) - diag(NaN(1,numel(prop))));
colorbar;
title('\bfcorrelations of d10');
xlabel({[prop{ord(32)},' (country #32) seems to be much less'], ...
    'synchronous than everyone else.'});
subplot(1,2,2);
imagesc(corr(d13(:,ord)) - diag(NaN(1,numel(prop))));
```

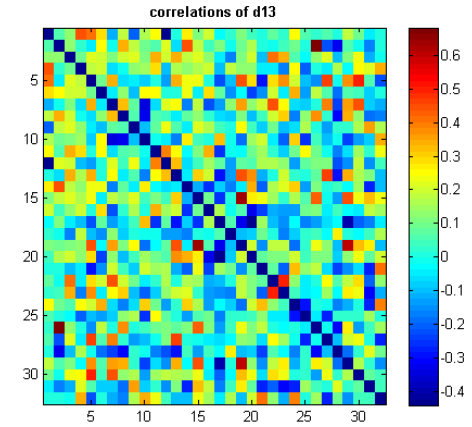
```
colorbar;
title('\bfcorrelations of d13');

report(['CONCLUSION: Some countries, Great Britan in particular, ', ...
    'have seasonal adjustments that are very different from those of ', ...
    'other countries in the sample. From the correlation plot we can ', ...
    'identify at least three groups of countries that behave ', ...
    'similarly. To find out more, we try to cluster them.']);
```

```
CONCLUSION: Some countries, Great Britan in particular, have seasonal
adjustments that are very different from those of other countries in the
sample. From the correlation plot we can identify at least three groups of
countries that behave similarly. To find out more, we try to cluster them.
```







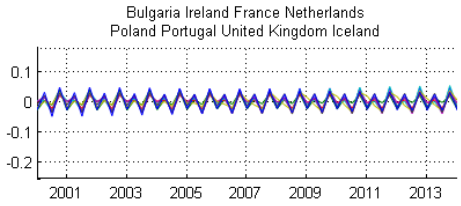


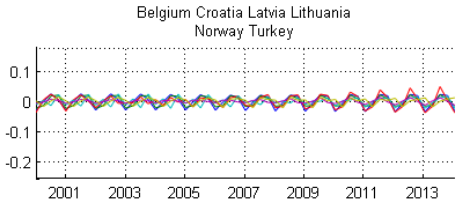
CLUSTER d10

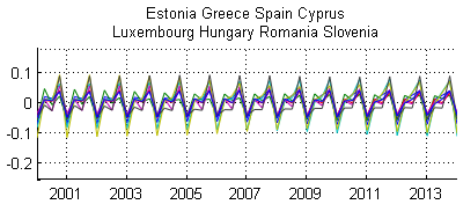
```
% % compute average seasonal factors
% sf = nan(4,size(d10,2));
% m = month(dates);
% for q = 1:4
%     pick = (m == q*3);
```

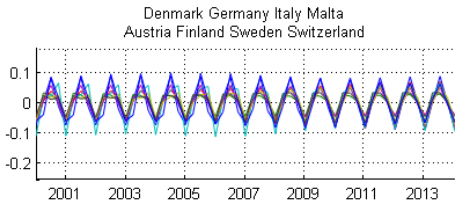
```
%     sf(q,:) = mean(d10(pick,:));
% end

% compute kmeans clusters
ncluster = 5;   % number of clusters; you can play around with this
[idx,sa] = kmeans(d10',ncluster);

% make a plot
figure('Position',size6, ...
    'Name',[name,': clustering the seasonal factors (d10)']);
nrows = ceil(sqrt(ncluster));
ncols = ceil(ncluster/nrows);
yl = [0,0];
% plot seasonal factors in one axis per cluster
colorOrder = get(gcf,'DefaultAxesColorOrder');
nColors    = size(colorOrder,1);
for c = 1:ncluster
    ax(c) = subplot(nrows,ncols,c);
    fidx = find(idx == c);
    colorRow   = 0;
    for cc = 1:numel(fidx)
        col = colorOrder(colorRow + 1,:);
        colorRow = mod(colorRow + 1, nColors);
        plot(ax(c),xgdp.(prop{fidx(cc)}),'d10','combined','logscale', ...
            'options',{'color',col})
    end
    ntit = ceil(numel(fidx)/4); ti = cell(1,ntit);
    for t = 1:ntit-1
        ti{t} = strjoin(ctry(fidx(1:4))');
        fidx(1:4) = [];
    end
    ti{ntit} = strjoin(ctry(fidx)');
    title(ti);
    grid on;
    axis tight;
    ylnew = ylim;
    yl(1) = min(yl(1),ylnew(1));
    yl(2) = max(yl(2),ylnew(2));
end
% same ylim for all axes (to ease comparison)
for c = 1:ncluster
    ylim(ax(c),[yl(1),yl(2)]);
end

report(['CONCLUSION: Surprisingly, the UK is not that special. ', ...
    'There is, however, great heterogeneity in the amplitudes of the ', ...
    'seasonal factors.']);
```

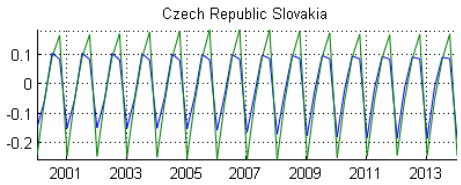
```
CONCLUSION: Surprisingly, the UK is not that special. There is, however,
great heterogeneity in the amplitudes of the seasonal factors.
```











finish up

```
disp(dline);

% turn warnings on again (or to whatever state they were)
warning(orig_warning_state);
```

==========

Published with MATLAB® R2014a

DEMO for X13 Toolbox: fixedseas

```
% turn warnings temporarily off
orig_warning_state = warning('off','all');

% line width
lwidth = 78;
% single and double line
sline = repmat('-',1,lwidth+1);
dline = repmat('=',1,lwidth+1);

% display with wrapped lines and leading space
report = @(s) disp(WrapLines(s,lwidth,' '));

% write heading
clc; disp(dline);
report(['DEMONSTRATION OF X-13 TOOLBOX FOR MATLAB : ', ...
    'run fixedseas and compare with output of X-13ARIMA-SEATS']);
report(['This script was developed with MATLAB Version ', ...
    '8.3.0.532 (R2014a)']);
disp(sline)
```

```
================================================================================
 DEMONSTRATION OF X-13 TOOLBOX FOR MATLAB : run fixedseas and compare with
 output of X-13ARIMA-SEATS

 This script was developed with MATLAB Version 8.3.0.532 (R2014a)

--------------------------------------------------------------------------------
```

Preliminaries

```
% Select type of trend

% LineSmoothing for Matlab up to 2014a
if verLessThan('matlab', '8.4')
    defaultOptions = {'LineSmoothing','on'};
else
    defaultOptions = cell(0);
end

% get correct path
p = fileparts(mfilename('fullpath'));   % directory of this m-file
% if the section is run with Shift-Ctrl-Enter ...
```

```
if isempty(p); p = [cd,'\']; end
% location of graphics files for use with X-13-Graph program
grloc = fullfile(p,'graphics\');

% size for figures with subplots
scSize = get(groot,'ScreenSize');
scWidth = scSize(3); scHeight = scSize(4);
sizeFig = @(h,v) round([0.04*scWidth, scHeight*(1-0.08-v/100)-50, ...
    h/100*scWidth, v/100*scHeight]);

size1 = sizeFig(40,80);
size2 = sizeFig(80,45);
size3 = sizeFig(95,45);
size4 = sizeFig(70,70);
size6 = sizeFig(75,68);
size8 = sizeFig(95,65);
size9 = sizeFig(95,90);

% line width
lwidth = 78;

% single and double line
sline = repmat('-',1,lwidth+1);
dline = repmat('=',1,lwidth+1);

% display with wrapped lines and leading space
report = @(s) disp(WrapLines(s,lwidth,' '));

% write heading
clc; disp(dline);
report(['DEMONSTRATION OF X-13 TOOLBOX FOR MATLAB : ', ...
    'demonstrating fixedseas']);
report(['This script was developed with MATLAB Version ', ...
    '8.3.0.532 (R2014a)']);
disp(sline);

% Here we select the type of the trend: 'ma', 'hp', 'detrend', 'spline'.
% 'polynomial' is possible, too, but does not work well with spr.
method = 'cma';
trans  = 'add';
```

```
================================================================================
 DEMONSTRATION OF X-13 TOOLBOX FOR MATLAB : demonstrating fixedseas

 This script was developed with MATLAB Version 8.3.0.532 (R2014a)

--------------------------------------------------------------------------------
```

Artificial Data

```
report(['We start our demonstration of fixedseas with some artificial data. ' ...
    'These data consist of a linear trend, two sinus cycles with ', ...
    'periodicity of 14 and 20, and some noise. We will try to identify ', ...
```

```
    'these periods (as if we didn''t know them already) and then filter ', ...
    'them out.']);

% make artificial data with trend, cycles, and noise
nobs = 500;

trend  = 0.025*(1:nobs)' + 5 - 0.00004*(1:nobs)'.^2;
cycle1 = 0.9 * sin((1:nobs)'*(2*pi)/20);
cycle2 = 1.0 * sin((1:nobs)'*(2*pi)/14);
cycle = cycle1 + cycle2;
resid  = 0.6 * randn(nobs,1);
data   = trend + cycle + resid;
% So we know that data has two periods, 14 and 20. We will try to find
% these periods.

t = clock;      % now
m = mod((t(2)-1:-1:t(2)-nobs),12)+1;
y = [0,-cumsum(diff(m)>0)] + t(1);
d = ones(size(y)) * 28;
dates = datenum(y,m,d);
dates = fliplr(dates)';

% We take a look at the data first.
s = x13([dates,data], x13spec);     % import into x13series just to
                                    % plot it nicely
plot(s);
title('Artificial Data');

figure; spr(data,method);
report(['The graph reveals clear spikes at 14 and 20. The spike at ', ...
    '28 is merely an echo of the one at 14, the one at 40 is an echo of ', ...
    'the one at 20, etc. We start by filtering out the shortest cycle ', ...
    '(at 14) first.']);

% filter out period 14
s = x13([dates,data],x13spec('fixedseas','period',14, ...
    'fixedseas','method',method,'fixedseas','transform',trans));
figure; spr(s.sa.sa,method,trans);
report(['The seasonally adjusted series shows no ', ...
    'spike at 14 anymore, but a clear spike at 20 (and possibly also ', ...
    'one are 40, but that is again merely an echo of the 20-cycle). ', ...
    'We now take out period 20 as well.']);

% filter out period 20 as well
s = x13([dates,data],x13spec('fixedseas','period',[14,20], ...
    'fixedseas','method',method,'fixedseas','transform',trans));
figure; spr(s.sa.sa,s.ir.ir,method,trans);
report(['The seasonally adjusted series and the residuals show no ', ...
    'spikes anymore. Possibly some cycles at high frequency (period = 2) ', ...
    'or at very low frequency (high values of period) are marked, but ', ...
    'these are not reliable and can be dismissed.']);

% add true values as variables
```

```
s.addvariable('ttr',dates,trend,'ttr',1,'true artificial trend');
s.addvariable('tsf',dates,cycle,'tsf',1,'true artificial cycle');
s.addvariable('tir',dates,resid,'tir',1,'true artificial noise');

% plot true vs estimated values
figure('Position',size1);
ax = subplot(3,1,1); plot(ax,s,'dat','sa','tr','comb');
ax = subplot(3,1,2); plot(ax,s,'sf','tsf','comb');
ax = subplot(3,1,3); plot(ax,s,'ir','tir','comb');

figure('Position',size2);
%
ax = subplot(1,2,1);
scatter(ax,cycle,s.sf.sf,'.b')
axis square; grid on;
xl = xlim; yl = ylim;
xl = [min(xl(1),yl(1)),max(xl(2),yl(2))];
xlim([xl(1),xl(2)]); ylim([xl(1),xl(2)]);
c = corrcoef(cycle,s.sf.sf);
title(['\bf',sprintf('seasonal factor (true vs estimated)\ncorrelation: %6.4f', ...
    c(1,2))]);
xlabel('true cycle');
ylabel('estimated seasonal factor');
%
ax = subplot(1,2,2);
scatter(ax, resid,s.ir.ir, '.b')
axis square; grid on;
xl = xlim; yl = ylim;
xl = [min(xl(1),yl(1)),max(xl(2),yl(2))];
xlim([xl(1),xl(2)]); ylim([xl(1),xl(2)]);
ok = ~isnan(s.ir.ir);
c = corrcoef(resid(ok),s.ir.ir(ok));
title(['\bf',sprintf('noise (true vs estimated)\ncorrelations: %6.4f', ...
    c(1,2))]);
xlabel('true errors');
ylabel('estimated residuals');

% Finally, we compare the diffrent trend extraction methods.
% Here we do not specity typearg, so we let fixedseas set a default.
% You can, of course, experiment with fifferent settings by adding
% 'fixedseas','typearg',parameter
sma = x13([dates,data],x13spec('fixedseas','period',[14,20], ...
    'fixedseas','method','cma', 'fixedseas','transform',trans, ...
    'series','name','Centered Moving Average'),'quiet');
shp = x13([dates,data],x13spec('fixedseas','period',[14,20], ...
    'fixedseas','method','hp', 'fixedseas','transform',trans, ...
    'series','name','Hodrick-Prescott'),'quiet');
spo = x13([dates,data],x13spec('fixedseas','period',[14,20], ...
    'fixedseas','method','poly', 'fixedseas','transform',trans, ...
    'series','name','Polynomial'),'quiet');
ssp = x13([dates,data],x13spec('fixedseas','period',[14,20], ...
    'fixedseas','method','spline', 'fixedseas','transform',trans, ...
    'series','name','Cubic Spline'),'quiet');
```

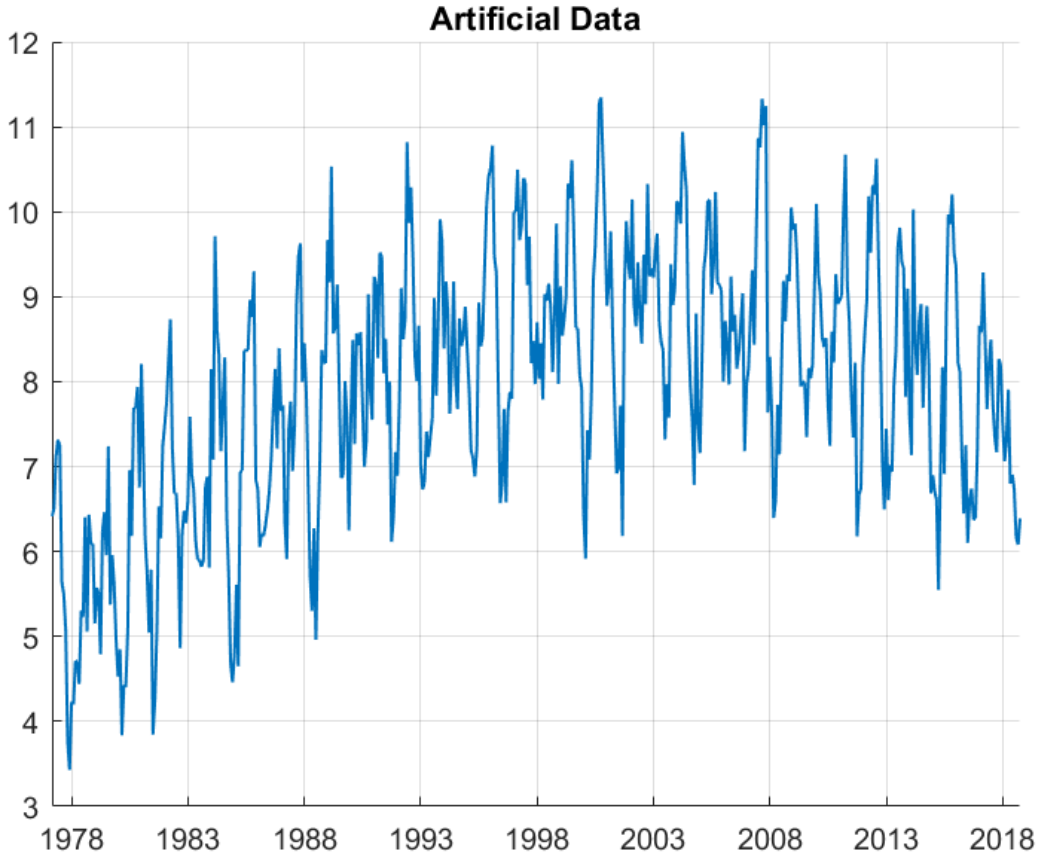
```
sdt = x13([dates,data],x13spec('fixedseas','period',[14,20], ...
    'fixedseas','method','detrend', 'fixedseas','transform',trans, ...
    'series','name','Detrend'),'quiet');

figure('Position',size2);
ax = subplot(1,2,1);
plot(ax,sma,shp,spo,ssp,sdt,'tr','comb');
ax = subplot(1,2,2);
plot(ax,sma,shp,spo,ssp,sdt,'sa','comb');
```

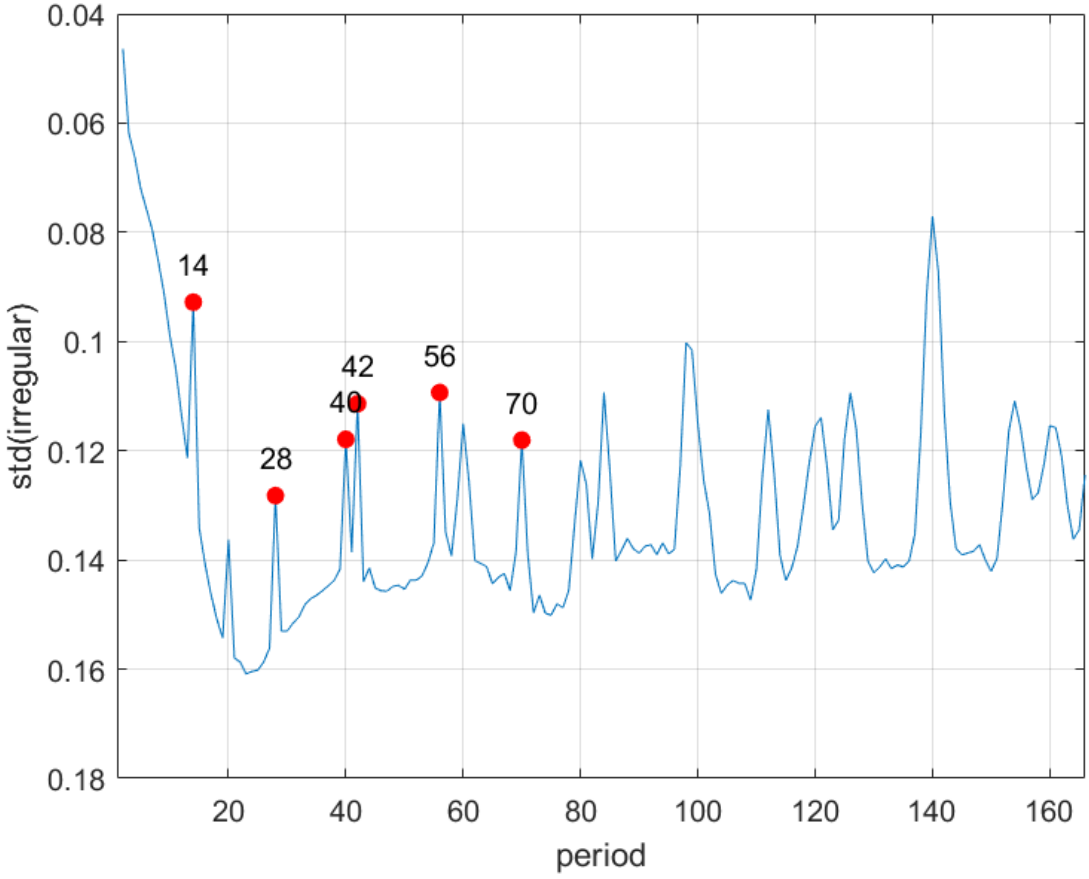
We start our demonstration of fixedseas with some artificial data. These data consist of a linear trend, two sinus cycles with periodicity of 14 and 20, and some noise. We will try to identify these periods (as if we didn't know them already) and then filter them out.

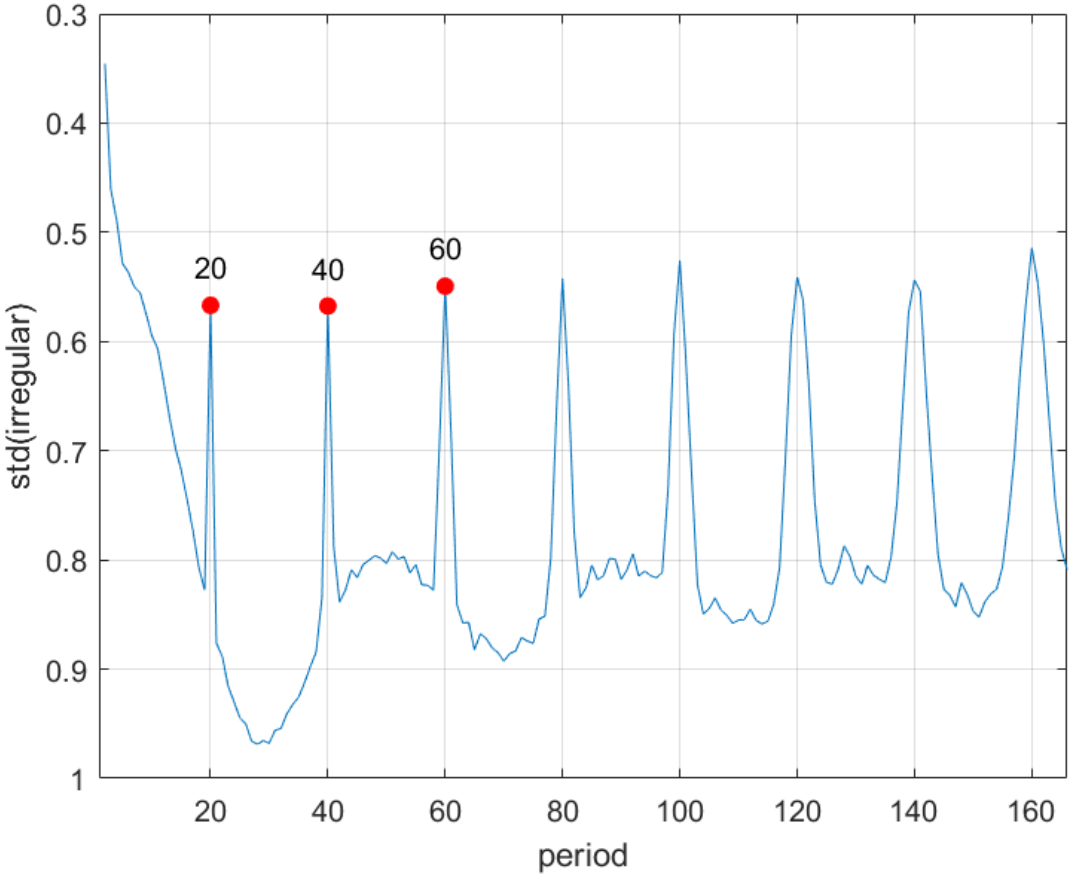
The graph reveals clear spikes at 14 and 20. The spike at 28 is merely an echo of the one at 14, the one at 40 is an echo of the one at 20, etc. We start by filtering out the shortest cycle (at 14) first.

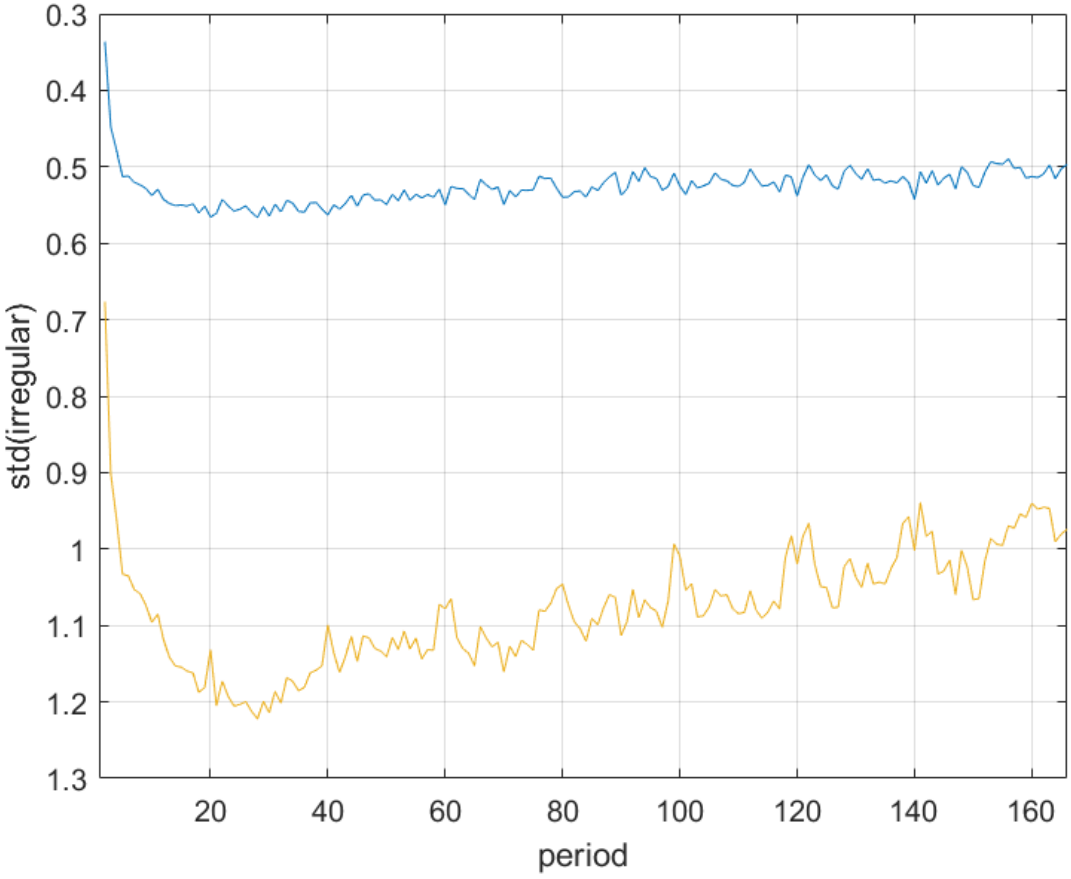
The seasonally adjusted series shows no spike at 14 anymore, but a clear spike at 20 (and possibly also one are 40, but that is again merely an echo of the 20-cycle). We now take out period 20 as well.

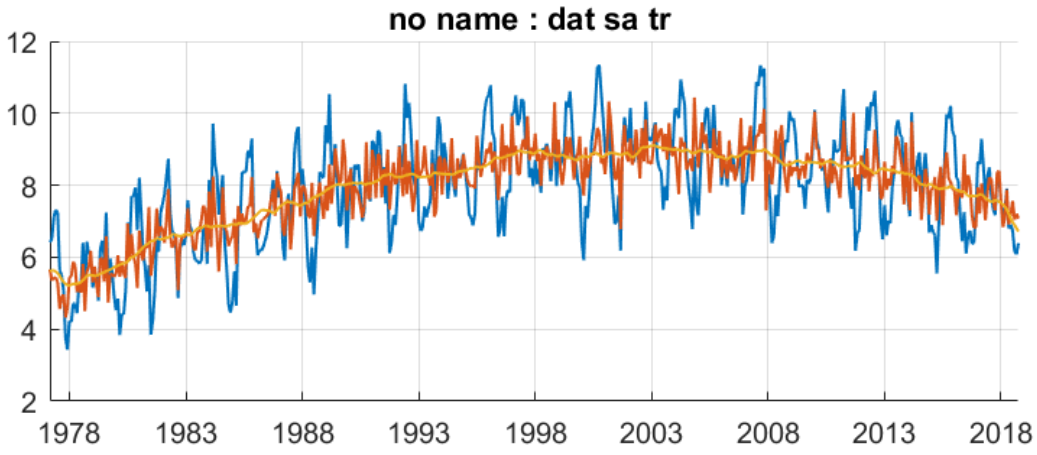
The seasonally adjusted series and the residuals show no spikes anymore. Possibly some cycles at high frequency (period = 2) or at very low frequency (high values of period) are marked, but these are not reliable and can be dismissed.

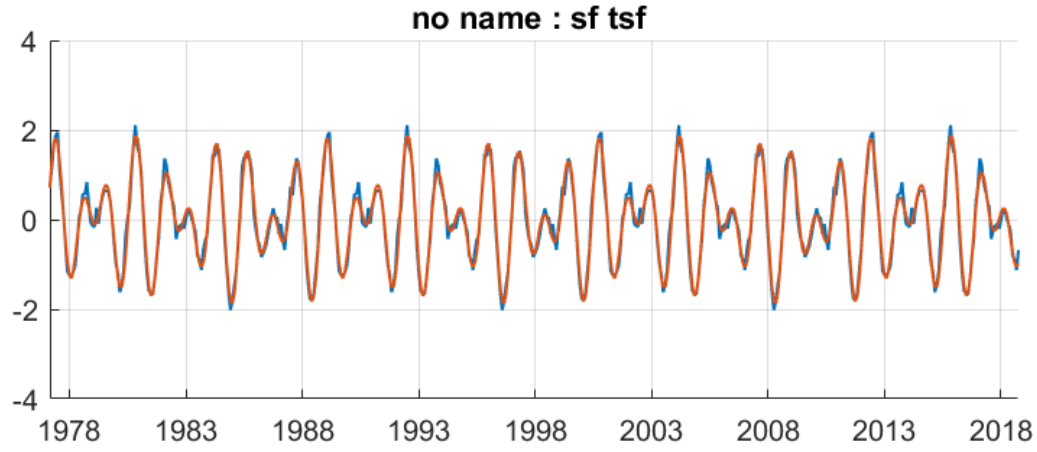

Artificial Data
12
11
10
9
8
7
6
5
4
3
1978
1983
1988
1993
1998
2003
2008
2013
2018

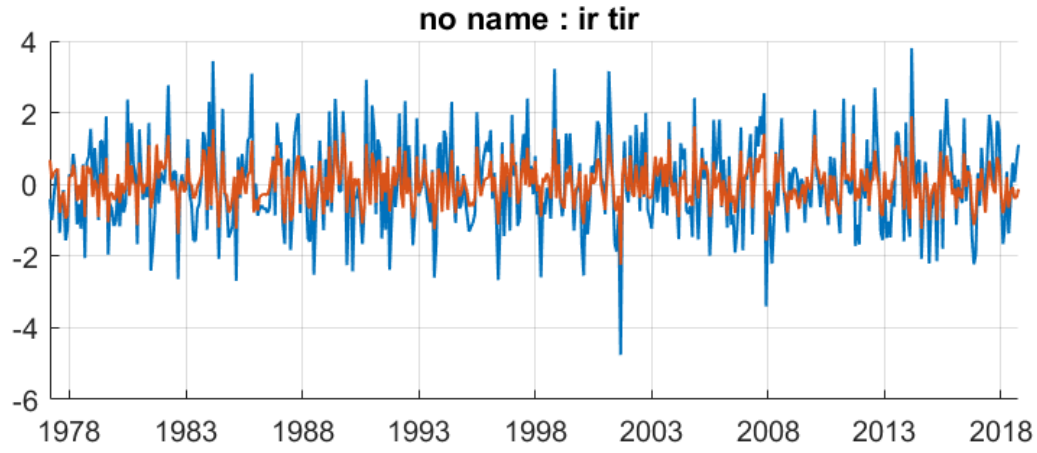

0.04
0.06
0.08
0.1
0.12
0.14
0.16
0.18
std(irregular)
20
40
60
80
100
120
140
160
period
14
28
40
42
56
70

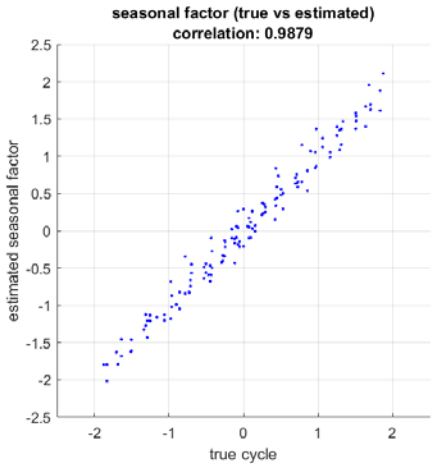

0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
std(irregular)
20
40
60
20
40
60
80
100
120
140
160
period

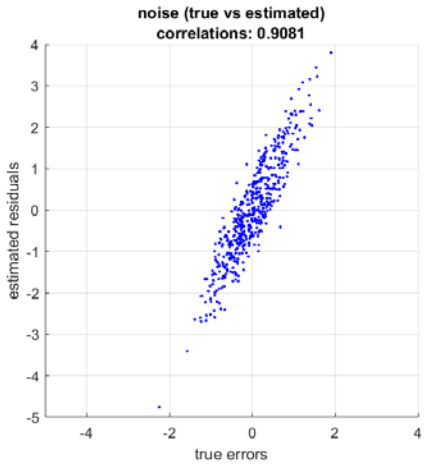

0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
1.1
1.2
1.3
std(irregular)
20
40
60
80
100
120
140
160
period

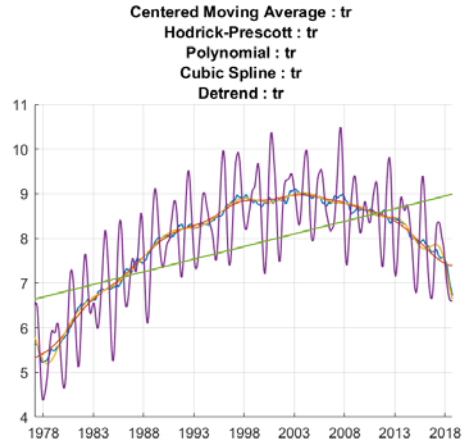

no name : dat sa tr
12
10
8
6
4
2
1978
1983
1988
1993
1998
2003
2008
2013
2018

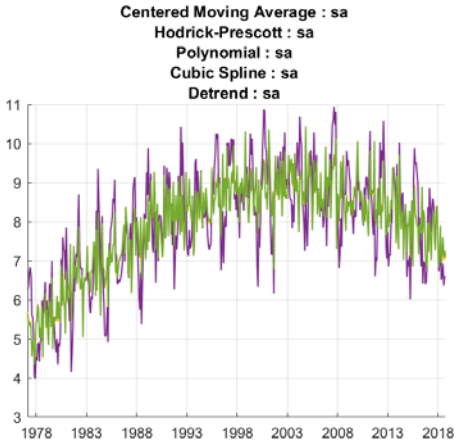

no name : sf tsf
4
2
0
-2
-4
1978
1983
1988
1993
1998
2003
2008
2013
2018


no name : ir tir
4
2
0
-2
-4
-6
1978
1983
1988
1993
1998
2003
2008
2013
2018









Real-Life Data

```
% US. Federal Highway Administration, Vehicle Miles Traveled
% [TRFVOLUSM227NFWA], retrieved from FRED, Federal Reserve Bank of St.
% Louis https://research.stlouisfed.org/fred2/series/TRFVOLUSM227NFWA/,
% December 31, 2014.
load(fullfile(p,'travel'));
name = 'Miles Traveled';

disp(sline)
fprintf(' We apply fixedseas to real data now.\n\n');
report(['Source and description of data: ',travel.source]);

% import into x13series just to plot it nicely
x = x13([travel.dates,travel.data], x13spec);
% A slightly more efficient way to get the data into an x13series object is
% this:
% x = x13series;
% x.addvariable('dat',travel.dates,travel.data,'dat',1);
% This method avoids calling the x13as program (since we don't want to
% compute anything right now).
plot(x);
```

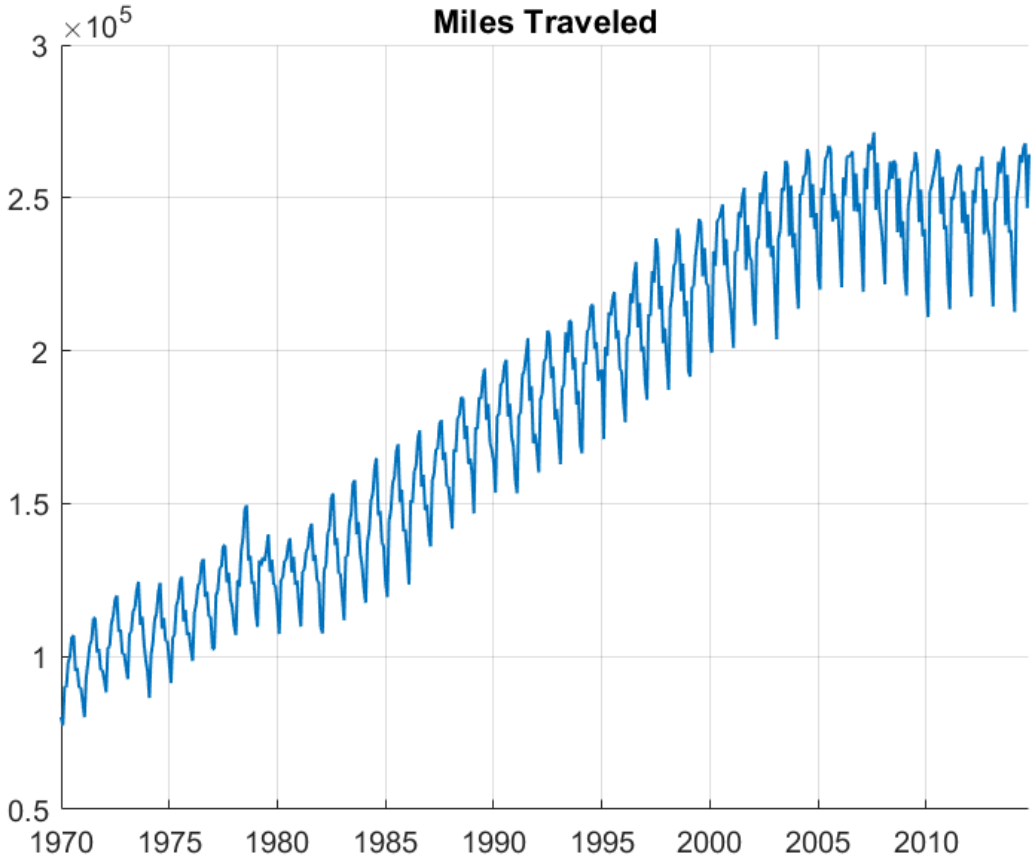
```
title(name);

figure; spr(travel.data,method);
report(['The graph reveals a clear spike at 12, and echos at ', ...
    'multiples of 12. These are monthly data with monthly ', ...
    'seasonality, which we filter out now.']);

fspec = x13spec('fixedseas','method',method);
% fspec = x13spec('fixedseas','method','detrend', ...
%     'fixedseas','typearg',datenum(2006,10,31));
x = x13([travel.dates,travel.data],fspec);
figure; spr(x.sa.sa,x.ir.ir,method);
report('The graph reveals no more spikes.');

report(['Let''s compare the result of fixedseas with the output of ', ...
    'X-13ARIMA-SEATS (see X13DemoTravel.m, whose result we replicate here).']);

% newspec = makespec(trans,'X11','DIAG', ...
%     'arima','model','(0 1 [1 4])(2 1 2)', ...
%     'regression','variables', ['(td easter[15] labor[8] thank[1] ', ...
%         'LS1979.May AO1993.May AO1995.Jan)'], ...
%     'regression','save','(td hol ao ls)', ...
%     'x11','seasonalma','s3x5', ...
%     'series','name',name);

xspec = x13spec( ...
    'transform'   ,'function'   ,'none', ...
    'arima'       ,'model'      ,'(0 1 [1 4])(2 1 2)', ...
       'regression'  ,'variables'  ,['(AO1993.May AO1995.Jan LS1979.May ', ...
        'td easter[15] labor[8] thank[1])'], ...
    'regression'  ,'save'       ,'(hol td ao ls)', ...
    'estimate'    ,'save'       ,'(mdl ref rsd rts est lks)', ...
    'check'       ,'save'       ,'(acf ac2 pcf)', ...
    'spectrum'    ,'save'       ,'(sp0 sp1 sp2 spr st0 st1 st2 str)', ...
    'x11'         ,'seasonalma' ,'s3x5', ...
    'x11'         ,'save'       ,'(d8 d10 d11 d12 d13 d16 e2 e3)', ...
    'series'      ,'name'       ,name);

% We instruct the program now to perform the X13ARIMA-SEATS computations
% (as defined in xspec) and the fixedseas computations (as specified in
% fspec).
x = x13([travel.dates,travel.data], x13spec(xspec,fspec));

% plot the result of fixedseas
figure('Position',size4);
ax = subplot(2,2,1); plot(ax,x,'dat','sa','tr','comb');
ax = subplot(2,2,3); plot(ax,x,'ir');
try
    ok = ~isnan(x.ir.ir);
    subplot(2,2,2); parcorr(x.ir.ir(ok),25);
    subplot(2,2,4); autocorr(x.ir.ir(ok),25);
catch
    report('(Sorry. Econometrics Toolbox is not installed.)');
```

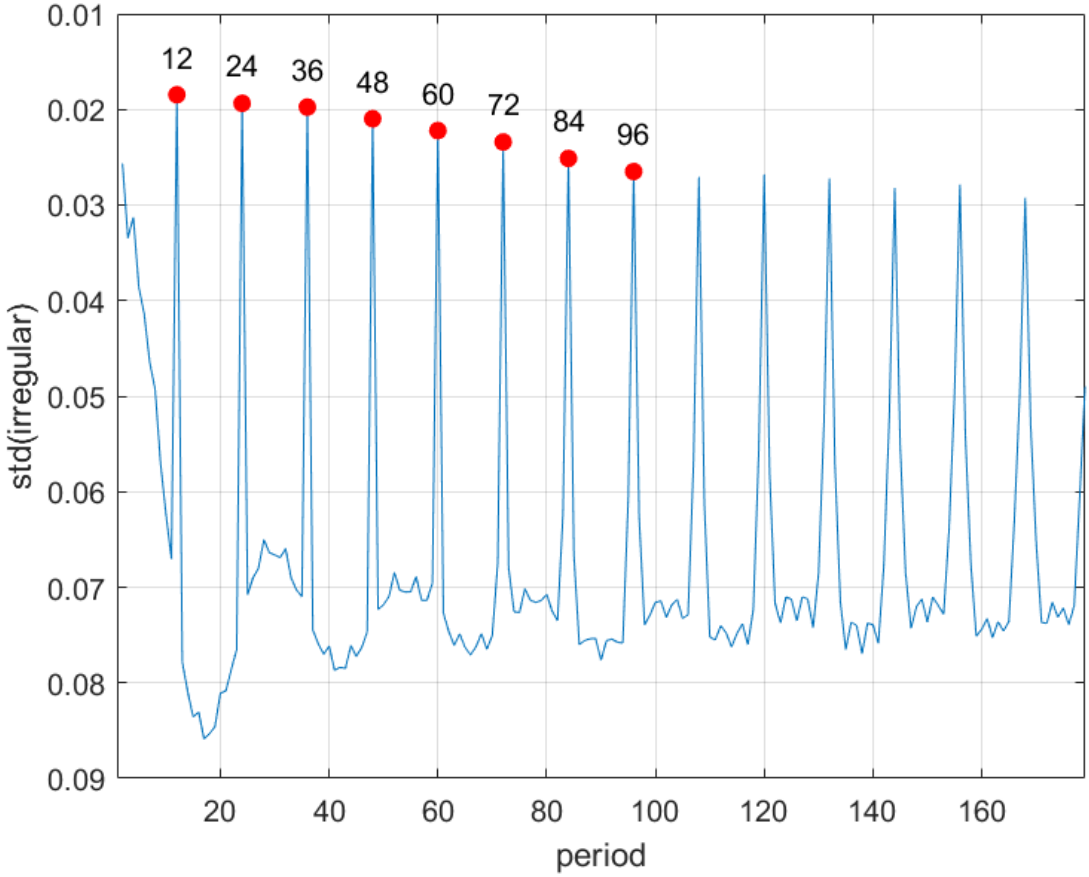
```
    subplot(2,2,2); spr(x.sa.sa,x.ir.ir,method);
    title('\bfSPR of .sa and .ir');
end
drawnow;
report(['If you have the econometrics toolbox, the graph shows the ', ...
    '(partial) autocorrelation function of the residuals. It is ', ...
    'obvious that significant correlation at period 12 remains, so ', ...
    'fixedseas was unable to remove all the seasonality.']);

% plot the result of X13ARIMA-SEATS
figure('Position',size4);
ax = subplot(2,2,1); plot(ax,x,'dat','d11','d12','comb');
ax = subplot(2,2,3); plot(ax,x,'rsd','d13','comb');
ax = subplot(2,2,2); plot(ax,x,'pcf');
ax = subplot(2,2,4); plot(ax,x,'acf');

% compare the two
figure('Position',size2);
ax = subplot(1,2,1);
plot(ax,x,'d12','tr','comb')
title('\bftrends')
ax = subplot(1,2,2);
plot(ax,x,'e2','sa','comb')
title('\bfseasonally adjusted series')
legend('X-13 (mod seas adj, e2)','fixedseas')
legend('Location','SouthEast')

% plot(x,'d12','tr','comb')

report(['Comparing the seasonally adjusted series (chart on the right) ', ...
    'of fixedseas with X-13ARIMA-SEATS shows that X-13ARIMA-SEATS does ', ...
    'a much better job of removing the seasonal pattern. Its trend ', ...
    '(chart on the left), however, is a little volatile.']);
```

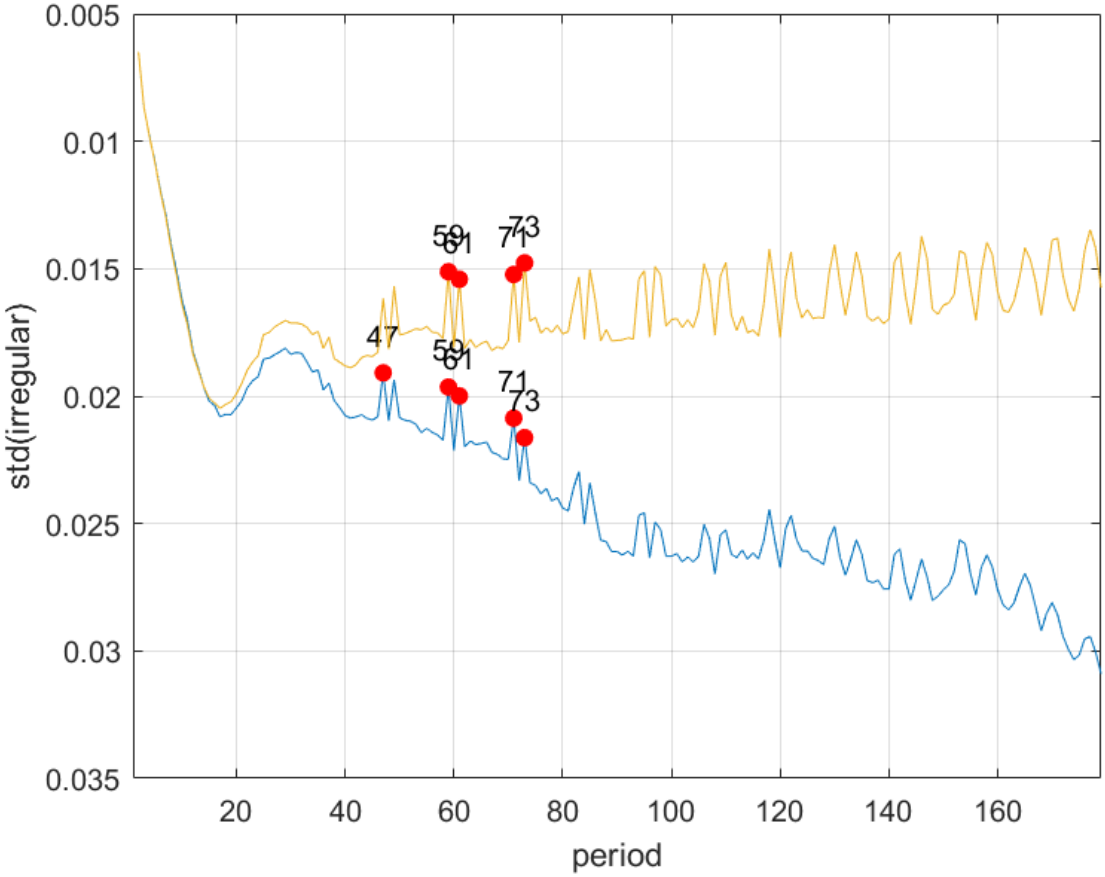
```
--------------------------------------------------------------------------------
 We apply fixedseas to real data now.

 Source and description of data: US. Federal Highway Administration, Vehicle
 Miles Traveled [TRFVOLUSM227NFWA], retrieved from FRED, Federal Reserve Bank
 of St. Louis https://research.stlouisfed.org/fred2/series/TRFVOLUSM227NFWA/,
 December 31, 2014.

 The graph reveals a clear spike at 12, and echos at multiples of 12. These
 are monthly data with monthly seasonality, which we filter out now.

 The graph reveals no more spikes.

 Let's compare the result of fixedseas with the output of X-13ARIMA-SEATS (see
 X13DemoTravel.m, whose result we replicate here).

 If you have the econometrics toolbox, the graph shows the (partial)
 autocorrelation function of the residuals. It is obvious that significant
 correlation at period 12 remains, so fixedseas was unable to remove all the
```

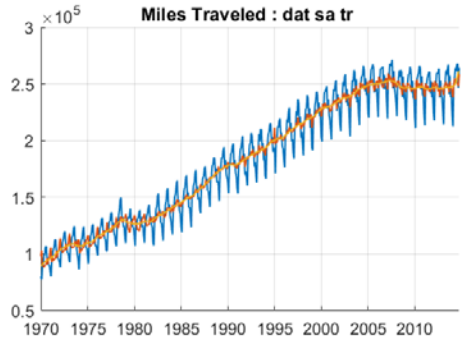
seasonality.

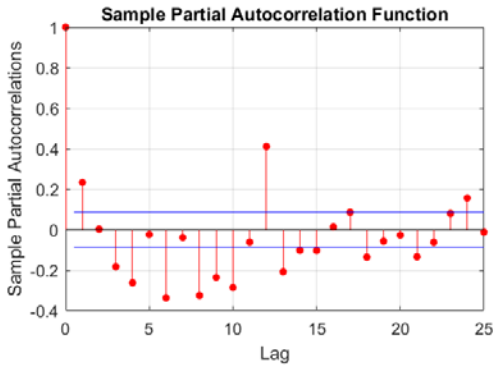
Comparing the seasonally adjusted series (chart on the right) of fixedseas with X-13ARIMA-SEATS shows that X-13ARIMA-SEATS does a much better job of removing the seasonal pattern. Its trend (chart on the left), however, is a little volatile.

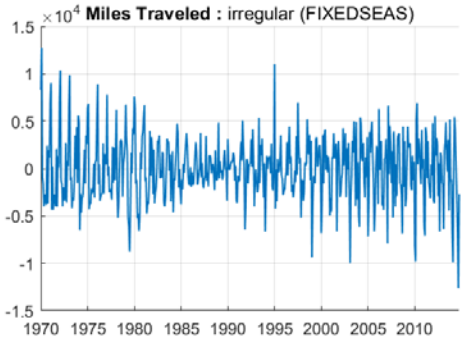


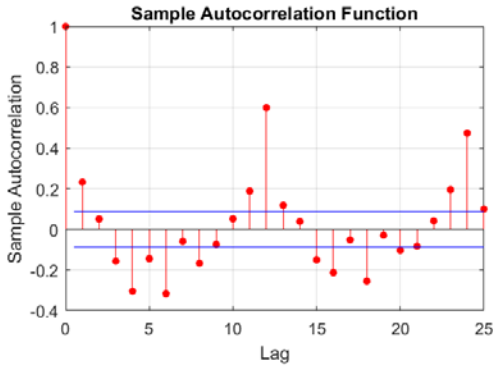

0.01
0.02
0.03
0.04
0.05
0.06
0.07
0.08
0.09
std(irregular)
12
24
36
48
60
72
84
96
20
40
60
80
100
120
140
160
period

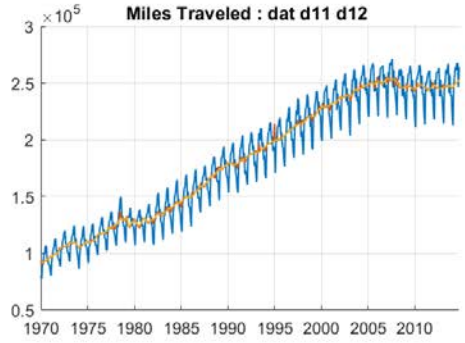

0.005
0.01
0.015
0.02
0.025
0.03
0.035
std(irregular)
20
40
60
80
100
120
140
160
period
47
59
61
71
73

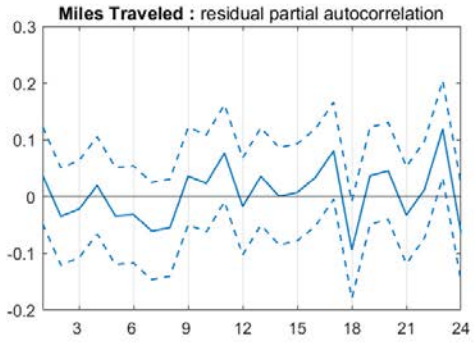

Miles Traveled : dat sa tr
×10^5
3
2.5
2
1.5
1
0.5
1970 1975 1980 1985 1990 1995 2000 2005 2010

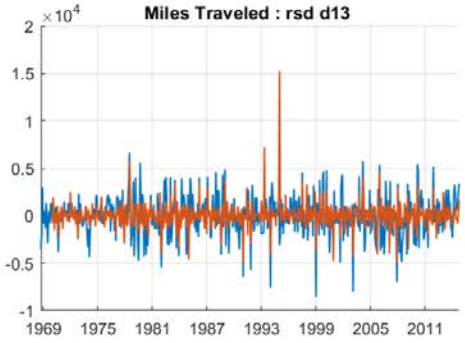

Sample Partial Autocorrelation Function
Sample Partial Autocorrelations
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
0 5 10 15 20 25
Lag

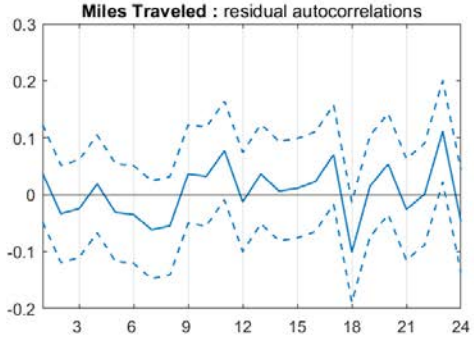

×10^4 Miles Traveled : irregular (FIXEDSEAS)
1.5
1
0.5
0
-0.5
-1
-1.5
1970 1975 1980 1985 1990 1995 2000 2005 2010

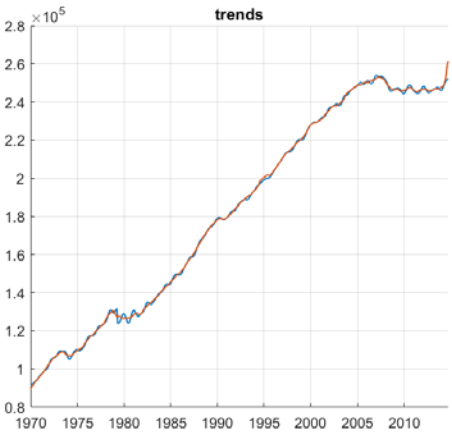

Sample Autocorrelation Function
Sample Autocorrelation
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
0 5 10 15 20 25
Lag

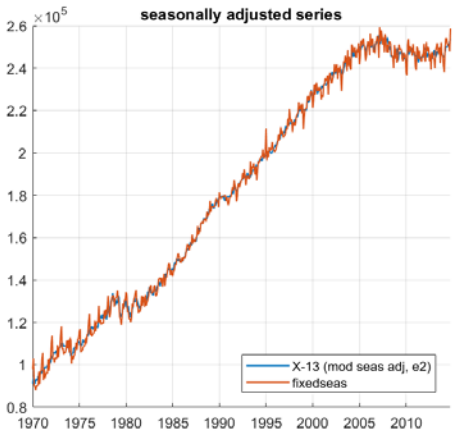

Miles Traveled : dat d11 d12
×10^5
3
2.5
2
1.5
1
0.5
1970 1975 1980 1985 1990 1995 2000 2005 2010


Miles Traveled : residual partial autocorrelation
0.3
0.2
0.1
0
-0.1
-0.2
3 6 9 12 15 18 21 24


Miles Traveled : rsd d13
×10^4
2
1.5
1
0.5
0
-0.5
-1
1969 1975 1981 1987 1993 1999 2005 2011


Miles Traveled : residual autocorrelations
0.3
0.2
0.1
0
-0.1
-0.2
3 6 9 12 15 18 21 24





finish up

```
disp(dline);

% turn warnings on again (or to whatever state they were)
warning(orig_warning_state);
```

==========

Published with MATLAB® R2017a

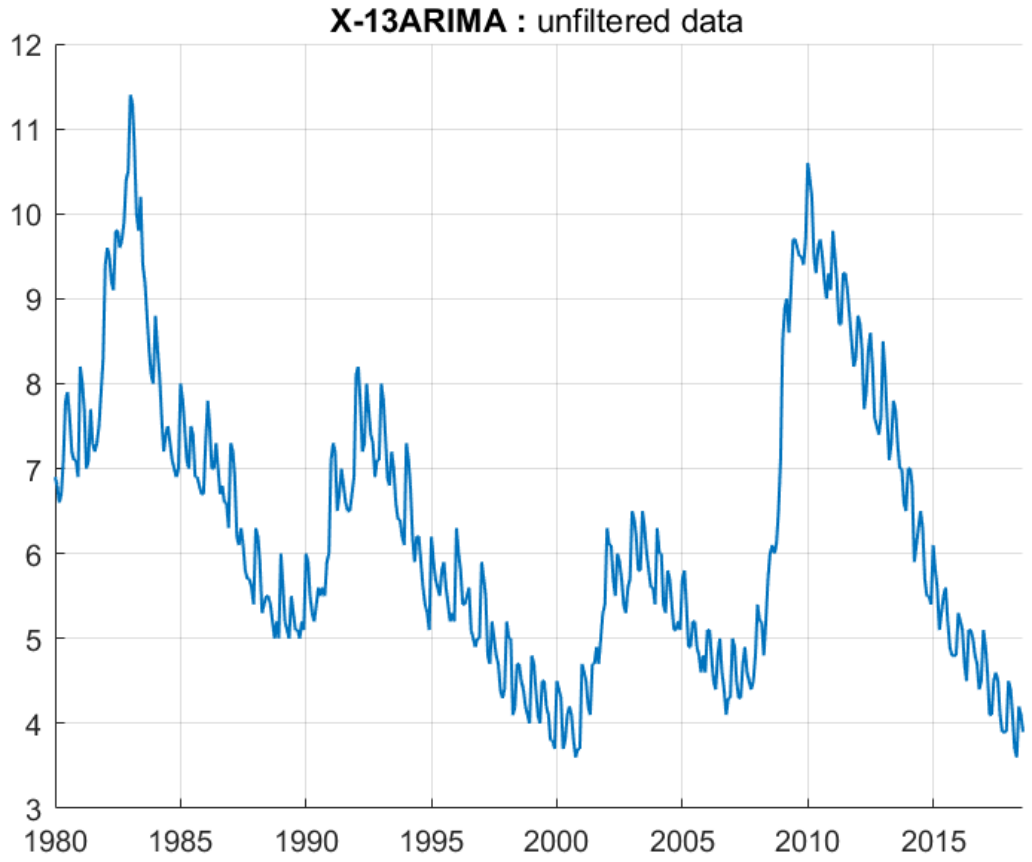
X13DEMOSEAS --- comparison of X-13, X-12, X-11, and seas

load the data and perform various seasonal adjustments

```
% U.S. Bureau of Labor Statistics, Civilian Unemployment Rate [UNRATENSA]
% retrieved from FRED, Federal Reserve Bank of St. Louis
load unemp

specadd = {'STOCK','ADD','NO OUTLIERS'};     % specs common to all runs

[~,xs] = seas([unemp.dates,unemp.data],12,'add');
x1 = x13([unemp.dates,unemp.data],makespec('X11','FIXED','CAMPLET', ...
    'series','title','simplified X-11',specadd{:}), 'x-11','-n');
x2 = x13([unemp.dates,unemp.data], ...
    makespec('DEFAULT','series','title','X-12',specadd{:}), 'x-12');
x3 = x13([unemp.dates,unemp.data], ...
    makespec('X','pickmdl','file','pure4.pml', ...
    'series','title','X-13ARIMA',specadd{:}));
x3s = x13([unemp.dates,unemp.data], ...
    makespec('S','SEATS','series','title','TRAMO-SEATS',specadd{:}));
```

look at the unfiltered data

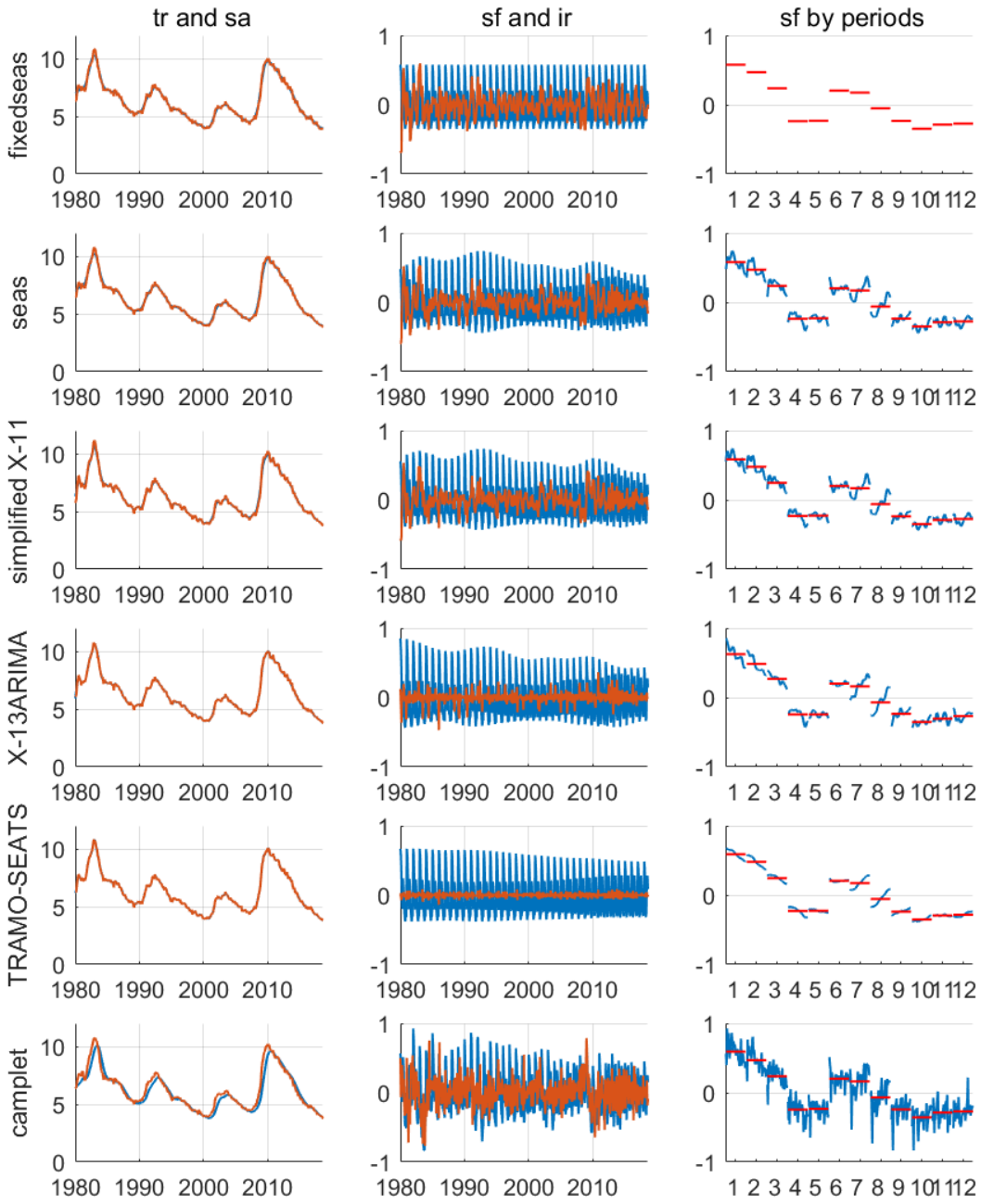
```
plot(x3);
```



results of different algorithms

```
figure('Position',[315 65 667 800]);
ticks = {'multdateticks',10};

ah = subplot(6,3,1); plot(ah,x1,'tr','sa','combined',ticks{:});
ylim([0,12]); title('\rm tr and sa'); ylabel('fixedseas');
ah = subplot(6,3,2); plot(ah,x1,'sf','ir','combined',ticks{:});
title('\rm sf and ir');
ah = subplot(6,3,3); plot(ah,x1,'sf','byperiod');
title('\rm sf by periods');

ah = subplot(6,3,4); plot(ah,xs,'tr','sa','combined',ticks{:});
ylim([0,12]); ylabel('seas'); title('');
ah = subplot(6,3,5); plot(ah,xs,'sf','ir','combined',ticks{:}); title('');
ah = subplot(6,3,6); plot(ah,xs,'sf','byperiod'); title('');

ah = subplot(6,3,7); plot(ah,x1,'d12','d11','combined',ticks{:});
ylim([0,12]); ylabel('simplified X-11'); title('');
ah = subplot(6,3,8); plot(ah,x1,'d10','d13','combined',ticks{:}); title('');
ah = subplot(6,3,9); plot(ah,x1,'d10','byperiod'); title('');

ah = subplot(6,3,10); plot(ah,x3,'d12','d11','combined',ticks{:});
```

```
ylim([0,12]); ylabel('X-13ARIMA'); title('');
ah = subplot(6,3,11); plot(ah,x3,'d10','d13','combined',ticks{:}); title('');
ah = subplot(6,3,12); plot(ah,x3,'d10','byperiod'); title('');

ah = subplot(6,3,13); plot(ah,x3s,'s12','s11','combined',ticks{:});
ylim([0,12]); ylabel('TRAMO-SEATS'); title('');
ah = subplot(6,3,14); plot(ah,x3s,'s10','s13','combined',ticks{:}); title('');
ah = subplot(6,3,15); plot(ah,x3s,'s10','byperiod'); title('');

ah = subplot(6,3,16); plot(ah,x1,'crp','csa', ...
    'combined','selection',[0 0 0 1 0 0 0 0 0 0 0 0 0],ticks{:});
ylim([0,12]); ylabel('camplet'); title('');
ah = subplot(6,3,17); plot(ah,x1,'csf','crp', ...
    'combined','selection',[0 1 0 0 0 0 0 0 0 0 0 0 0],ticks{:});
title('\rm sf and ir'); title('');
ah = subplot(6,3,18); plot(ah,x1,'csf','byperiod');
title('\rm sf by periods'); title('');
```



conclusion

```
disp(WrapLines(['X-11 is not the original version of the X-11 algorithm. ', ...
    'But the seasonal factors are comparable to X-13 (except for the ', ...
    'edge of the sample, because the simplified X-11 does not use ARIMA ', ...
```

```
    'to extend the data). The same is true for seas.'],63));

disp(WrapLines(['A difference, however, is the irregular component. ', ...
    'This is supposed to contain only high frequencies, but in X-11, ', ...
    'seas, and fixedseas, the irregular contains some lower frequency ', ...
    'as well.'],63));

disp(WrapLines(['camplet also produces quite a different result. ', ...
    'First of all, the trend is backward looking and has therefore a ', ...
    'phase shift. Moreover, the seasonal factors are quite volatile with ', ...
    'this algorithm.'],63));
```

```
X-11 is not the original version of the X-11 algorithm. But the
seasonal factors are comparable to X-13 (except for the edge of
the sample, because the simplified X-11 does not use ARIMA to
extend the data). The same is true for seas.

A difference, however, is the irregular component. This is
supposed to contain only high frequencies, but in X-11, seas,
and fixedseas, the irregular contains some lower frequency as
well.

camplet also produces quite a different result. First of all,
the trend is backward looking and has therefore a phase shift.
Moreover, the seasonal factors are quite volatile with this
algorithm.
```

Published with MATLAB® R2017a