X-13 Toolbox for Matlab

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

May 24, 2021, Version 1.50

Please reference the software as

> Yvan Lengwiler, 'X-13 Toolbox for Matlab, Version 1.50', Mathworks File Exchange, 2021, 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 57 of the X-13ARIMA-SEATS program 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 X-13 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

Basic Procedures

The toolbox is an interface that makes it relatively easy to use the X-13ARIMA-SEATS program from the US Census Bureau with Matlab. This part of the documentation describes the basic procedures you will use whenever you use this toolbox.

Before going into any detail, note that there is a graphical user interface called **guix** that allows you to do (almost) anything the toolbox can do. The interface is self-explanatory as far as possible (there are many options, so some knowledge of the process of seasonal adjustment will be helpful), so if you do not like reading the documentation, just fire up **guix** and use the toolbox that way.

. . .

Still here? Ok, so let's explain the basic ingredients.

The toolbox relies on three custom data classes, **x13spec**, **x13series**, and **x13composite**. **x13spec** contains all the specifications that instruct the X-13ARIMA-SEATS program about what it should do. For instance, it specifies the algorithm that is used for performing the seasonal adjustment and which variables and tables should be generated.

The Census program can do two types of adjustments. The first type is an adjustment to a single time series. Doing this produces a variable of class **x13series**. The second is an adjustment to a whole collection of timeseries, that also adjusts the aggregate of these series. X-13 calls this a composite, and the Matlab toolbox creates a variable of class **x13composite** to hold such a seasonal adjustment.

If you want to work on the command-line instead of using **guix**, or wish to use the toolbox in a script you develop, you first need to create a specification. You can do this with **x13spec** directly. You can also use **makespec**, which is a bit more convenient. After that, **x13** can be used to invoke the Census program.

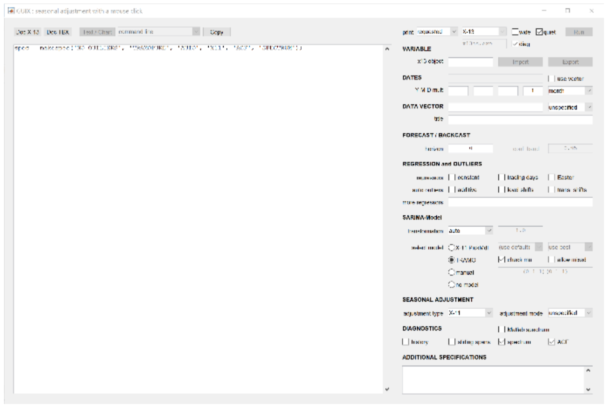
Finally, there is a method, **InstallMissingCensusProgram** that is called in the background whenever a needed component is missing. It attempts to install that component on the fly. However, you can also manually install pieces of software using this method.

We now provide the documentation of **guix**, **makespec**, **x13**, and **InstallMissingCensusProgram**. Not many users will need more than that.

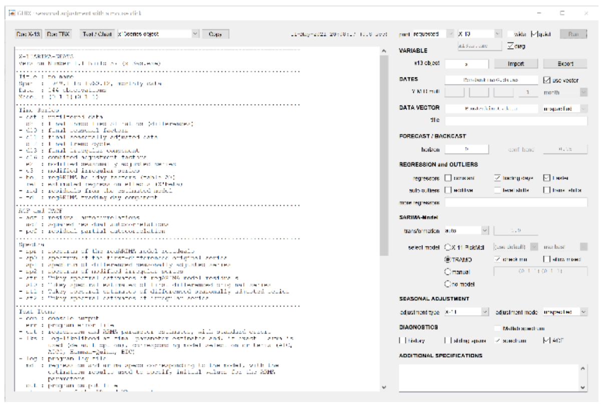
guix

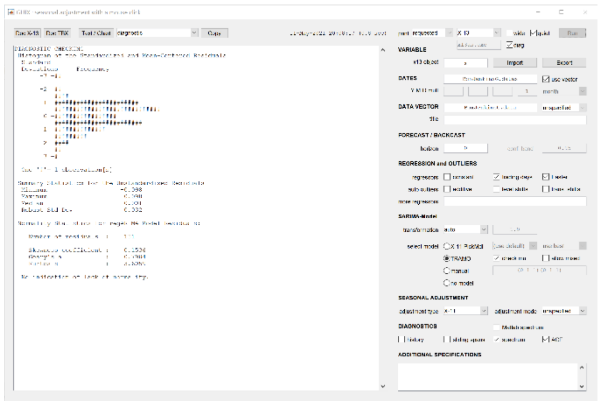
```
guix is a graphical user interface that allows you to easily create a
specification for a seasonal adjustment, perform the computations, and
view the results in the GUI. guix supports only single time series
(x13series), not composites (x13composite).

Usage:
  guix
  guix('style')
  guix 'style'
  guix('variable')
  guix('variable','style')
  guix variable
  guix variable 'style'
  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). X-13 requires a vector of
observation dates and the corresponding data. Dates can be specified by
entering the start date and the period (e.g., monthly). Alternatively, it
is also possible to use a vector of datenums. The data that are to be
worked on are given by a vector of floats. The data vector and, if used,
dates vector, 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.

NOTE: This GUI was programmatically created, without GUIDE. Other than
the code generated with GUIDE, it uses nested functions, which has the
```

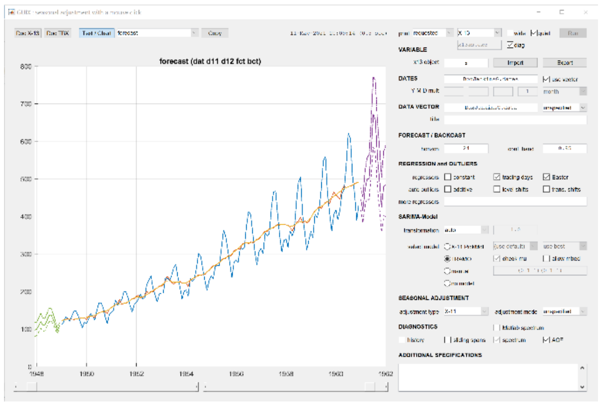
advantage that all variables that are defined in the main function are also in the scope of the nested functions. Moreover, the code is more transparent.



```
spec = makespec('NO OUTLIERS', 'TRAMOPURE', 'AUTO', 'X11', 'ACF', 'SPECTRUM');
```







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 those 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','TDAYS','DIAG'}
  'X'       is equal to {'AUTO','PICKBEST' ,'X11'  ,'AO','TDAYS','DIAG'}
  'S'       is equal to {'AUTO','TRAMOPURE','SEATS','AO','TDAYS','DIAG'}
If no argument is used (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 accommodate all needs, unless the
shortcuts are as detailed as the original specification possibilities,
which would defy their purpose. 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','TRAMOPURE','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).

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(..., 'x-13')
  x = x13(..., 'html')
  x = x13(..., 'x-12')
  x = x13(..., 'x-11')
  x = x13(..., 'method1') or x = x13(..., 'method I')
  x = x13(..., 'camplet')
  x = x13(..., 'fixedseas')
  x = x13(..., 'prog',filename)
  x = x13(..., 'progloc',path)
  x = x13(..., 'quiet')
  x = x13(..., '-s', '-c', '-w', ... etc)
  x = x13(..., 'noflags')
  x = x13(..., 'graphicsmode')
  x = x13(..., 'graphicsloc',path)
  x = x13(..., 'fileloc',path)

'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.

Four 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.

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.

The 'x-11' switch uses an approximate version of the original Census X-11 algorithm from 1965. The original Census X-11 program is available, but is not compatible with this toolbox because the format of its in- and output is very different from X-12 and X-13. Instead, an approximate version of X-11 is implemented in Matlab and this is used when you set the 'x-11' switch. This has many limitations and its use may be limited. It has one important advantage, however, in that the simplifiex X11 implementation can deal with arbitrary frequencies (so not only monthly or quarterly data as the Census programs do). See help x11 for further information.

Finally, 'method1' is similar to 'x-11', but uses an approximate version of the original Method I argorithm that was developed by

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.

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. It is preferrable to use the 'quiet' switch instead, because then the messages are not shown on the console (as would be the case with '-q'), but they are still available as messages stored in the x13series object.

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. 'prog' is the name of the executable. By default, this is 'x13as', or 'x12a' (or 'x12a64' on a 64-bit computer) if the 'x-12' option is set. But it is also possible to specify a m-file that perfoms the necessary seasonal adjustment computations.

'progloc' indicates the folder where the executables can be found. In the argument following 'progloc', the term '%tbx%' is replaced by the root directory of this toolbox. By default, 'progloc' is '%tbx%\exe', i.e. the 'exe' subdirectity of the X-13 toolbox. If 'prog' is an m-file then the default 'progloc' is '%tbx%\seas'.

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.

Another application is to use a seasonal adjustment algorithm in Matlab's
language. seas.m is an example or this. By setting
  x = x13(..., 'prog','seas.m'),
you use this self-made seasonal adjustment algorithm in place of the
Census programs. The advantage is that you have more freedom to
experiment. Also, you are not constrained to mopnthly or quarterly
frequencies.

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 X-11 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.

InstallMissingCensusProgram

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

  *** NORMAL OPERATION

  Normally, x13.m will download and install any missing piece of software
  from the US Census Bureau as soon as it is required. You can, however,
  also install such software and documentation "manually" by invoking
  InstallMissingCensusProgam.

  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
    'campletdoc'    the original working paper presenting the CAMPLET
                    algorithm
  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-b57.zip' from US Census Bureau website ... success.
   Downloading 'x13ashtml-v1-1-b57.zip' from US Census Bureau website ... success.
   Downloading 'docsx13.zip' from US Census Bureau website ... success.
   Downloading 'docsx13acc.zip' from US Census Bureau website ... success.
   Downloading 'dt9628e.pdf' from Banco d'Espagna 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.
   Downloading 'ffc2000.pdf' from SAS Institute website ... success.
   Downloading '25_2015_abeln_jacobs.pdf' from Australian National University
website ... success.
   *** 12 of 12 requested packages installed. ***
```

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

*** IF IT DOES NOT WORK

Normally, this program will download the specific pieces of software and documentation that are provided by the U.S. Census Bureau and copy them to the appropriate locations. This assumes, however, that your computer allows you to download and run software from the internet. This may not apply to a professional environment where IT security issues are managed centrally. If you cannot download and run externally-acquired software, you will need help from an IT administrator at your workplace.

Also, this utility works only with Windows computers. If you use a different operating system, you will have to download and place the necessary files manually.

You find all the files you need at the the U.S.Census website (https://www.census.gov/data/software/x13as.X-13ARIMA-SEATS.html). Search in the different ZIP available there to locate the correct files. Also, it may be possible to obtain the source code of the X-13 program so that you may be able to compile this for so far unsupported operating systems. (If you do that, please let me know; I would be interested about this.)

If all usable files are present, the exe sub-direcory should contain the following files:
```
 (**) x13as.exe
      x13ashtml.exe
  (*) x12diag03.exe
      x12a64.exe
      x12a.exe
  (*) libjpeg.6.dll
  (*) libpng3.dll
  (*) zlib1.dll
```
In addition, the doc sub-directory should contain
```
  (*) docX13AS.pdf
      docX13ASHTML.pdf
```
Only the double-starred file is essential. The single-starred files are the documentation and the files needed for a small addition (which is, incidentally, used by X13DemoComposite.m). The unstarred files give you access to the vintage X-12 version.

*** OTHER THINGS YOU CAN DOWNLOAD WITH THIS UTILITY
    (these are of only marginal interest)

In addition, five other programs that are related to X-13ARIMA-SEATS can also be downloaded:

```
    'x13graph'      The JAVA version of the X-13 graph program.
    'genhol'        A program that allows the user to create variable files
                    for holidays.
    'x11prog'       An early version of the Census program.
    'x11doc'        Some documentation of the early X-11 version.
    'winx13'        Windows version of the X-13ARIMA-SEATS 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.

  These programs are not directly supported by the Matlab-Toolbox. The
  x13graph java program can be used if you add the 'graphicsmode' switch
  when calling x13, but you need to start the graph program outside of
  Matlab. Likewise, genhol, or the version with a GUI called wingenhol, is
  not used by the toolbox directly. You can create holiday variable files
  with it, and then use these files with the toolbox. But the interaction
  with genhol does not happen from within the toolbox. X-11 is not
  supported because its syntax is completely different from X-13ARIMA-SEATS
  and it is potentially prone to Y2K problems. The remaining programs
  winx13, x13data, cnv, and sam have no clear use for users of the toolbox.
  The download option provided here is only for completeness and may be
  useful for users who interact with the Census program also outside of
  Matlab.

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

  Using this function with the 'all' argument yields
   Downloading 'x13as-v1-1-b57.zip' from US Census Bureau website ... success.
   Downloading 'x13ashtml-v1-1-b57.zip' from US Census Bureau website ... success.
   Downloading 'docsx13.zip' from US Census Bureau website ... success.
   Downloading 'docsx13acc.zip' from US Census Bureau website ... success.
   Downloading 'dt9628e.pdf' from Banco d'Espagna 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.
   Downloading 'ffc2000.pdf' from SAS Institute website ... success.
   Downloading '25_2015_abeln_jacobs.pdf' from Australian National University
website ... success.
   Downloading 'x11.zip' from EViews(R) website ... success.
   Downloading 'ShiskinYoungMusgrave1967.pdf' from US Census Bureau website ...
success.
```

```
Downloading 'x11_french.pdf' from US Census Bureau website ... success.
Downloading 'x11_spanish.pdf' from US Census Bureau website ... success.
Downloading '1980X11ARIMAManual.pdf' from US Census Bureau website ... success.
Downloading 'Emanual.pdf' from US Census Bureau website ... success.
Downloading 'genhol_V1.0_B9.zip' from US Census Bureau website ... success.
Downloading 'wingenhol-v1.0-B3.zip' from US Census Bureau website ... success.
Downloading 'winx13-v3.0.zip' from US Census Bureau website ... success.
Downloading 'X13GraphJava_V3.0.zip' from US Census Bureau website ... success.
Downloading 'X13GraphJavaDoc.pdf' from US Census Bureau website ... success.
Downloading 'x13data-v2-0.zip' from US Census Bureau website ... success.
Downloading 'X13sam-v1.1.zip' from US Census Bureau website ... success.
Downloading 'toolsx13.zip' from US Census Bureau website ... success.
*** 26 of 26 requested packages installed. ***
```

x13spec class

x13spec is a class that contains a specification for a run of the X-13ARIMA-SEATS program. An x13spec can be created directly with the `x13spec` function, or, more conveniently, with `makespec`, see above. An x13spec variable has several methods that can be applied to it. The most important being `display` (short form display of the content) and `disp` (long form display of the content). There is also a method `str = displaystring(spec)` (where spec is an x13spec variable) that stores the long form display to a string.

x13spec

```
x13spec is the class definition for x13spec objects. Such an object is
used to set all specifications of a run of the X-13ARIMA-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','title','rainfall','transform','function','auto')
would set title = 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{
      title = 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','title','rainfall','x11','save','d10');
  spec2 = x13spec(spec1,'series','title','snowfall');
then spec2 contains save = d10 in the x11-section (inherited from spec1),
but title = snowfall in the series-section (the title rainfall was
overwritten).

Remark 3: accumulating keys -------------------------------------------
The keys 'save','savelog','print','variables','aictest','types','user',
'usertype','keys','values','smoothmethod','methodarg' behave differently.
These keys are accumulated,
  spec = x13spec('x11','save','d10');
  spec = x13spec(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
AddRequests and SaveRequests (to overwrite keys) methods.
```

Multiple entries can be added to an accumulating key using different, equivalent syntaxes,

```
  x13spec('x11','save','d10 d11 d13');
  x13spec('x11','save','(d10 d11 d13)');
  x13spec('x11','save',{'d10','d11','d13'});
```

Remark 4: creating 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 [] in place of the key, i.e. if spec has an 'x11' section, then spec = x13spec(spec,'x11',[],[]) removes the 'x11' section completely from this spec.

To remove a key from a section, use an [] as value, as follows:
spec = x13spec('x11','save','d10','x11','savelog','q') produces

```
  x11{
      save = d10
      savelog = q
  }
```

Then spec = x13spec(spec,'x11','save',[]) removes the 'save' key and produces

```
  x11{
      savelog = q
  }
```

spec = x13spec(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.enforce(prog), where prog is the name of one
of the US Census Bureau seasonal adjustment executable files, or a custom
m-file that performs a seasonal adjustment. The procedure removes items
from the spec that are not compatible with the program that is specified.
Normally the user does not have to deal with this, as the .enforce method
is automatically applied by x13.m whenever needed.

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')
will cause problems, so avoid it.

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 and custom sections ---------------
x13spec also accommodates three sections that have no meaning for the x13as program. These sections are 'fixedseas', 'camplet', and 'custom'. The contents of these sections are not transmitted to x13as. Instead, they are passed to separate Matlab programs (fixedseas.m or camplet.m or a custom program specified in the x13 call by 'prog','name.m').

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).
- 'mode' 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.
- 'save'      This is the list of variables that should be saved.
              Possible values are ''tr', sa', 'sf', 'ir', and 'si'.
- '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.

camplet computes a seasonal adjustment proposed by Abeln and Jacobs, 2015.
The results are embedded into the x13series object as variables 'sf'
(seasonal factor), 'sa' (seasonally adjusted), as well as a couple of
series unique to this algorithm (seee 'help camplet'). 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.

The 'camplet' section supports the following keys:
- '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.
- 'save'      This is the list of variables that should be saved.
              Possible values are 'sa', 'sf', 'bar', 'fcs', 'fer', 'rer',
              'gra', 'g', 'nol', 'psh', 'cca', 'ca', 'm', 'lle', 't'.

- 'options'
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.
If set by the user, these are stored in the 'options' key of the struct.

Note that camplet.m does an additive or a log-additive decomposition of
the data. This choice is not stored in the camplet-transform or
camplet-mode keys (which both do not exist), but rather in the
transform-function key (outside of the camplet section).

You can also define your own, custom m-file that performs a seasonal
adjustment. You have to observe a few restrictions on the form of your
output, so that it is readable for the toolbox. seas.m in the seas
subfolder is an example of such a custom algorithm.

The 'custom' section supports the following keys: 'period', 'mode',
'save', 'options'. What these mean and which values are legit depends on
the custom m-file you use, and whose name is stored on the .prog property
of the x13series object. The seas.m file is an example of such a custom
file. It accepts in the 'mode' section either 'none', 'add',  'logadd',
or 'mult'. In the 'save' section it accepts the same as fixedseas. seas.m
does not use for the 'period' and 'options' sections (but your own m-file
could use these sections).

--------------------------------------------------------------------------
PROPERTIES and METHODS

The resulting spec-object contains one property for each section entered.
In addition, it also has the following properties:
- isempty         boolean     spec.isempty returns true if the spec
                              contains no sections.
- isComposite     boolean     True if 'composite' is one of the series.
- adjmethod       char        The name of the method used for seasonal
                              adjustment, e.g. 'x11', 'seats',
                              'fixedseas', ...
- transfunc       char        The function stored in
                              spec.transform.function. If that is
                              missing, the content of
                              spec.transform.power is mapped to either
                              'none' or 'log' (or noting, if no mapping
                              is sensible).
- adjmode         char        Typically 'add' or 'logadd' or 'mult'.
- requesteditems  cells       All the variables requested for saving
                              anywhere in the spec (with the 'save' key).
- mainsec         char        Either 'series' or 'composite', depending
                              on whether the spec belongs to composite
                              data or not.

Several methods are available with a x13spec. Most of them are mainly used
```

```
internally, and called by makespec.m, x13series.m, or x13.m, but they
could also be used by a user:

spec.addtriplet(section,key,value) adds one section-key-value entry.
- merge                 spec3 = spec1.merge(spec2) makes spec3 that contains
                        the contents of spec1 and spec2.
- specminus             spec3 = spec1.specminus(spec2) removes all
                        components in spec1 that are also present in spec2
                        and places the remainder into spec3.
- copy                  obj2 = obj.copy creates an exact, but intependent
                        copy of obj. Note that simply assigning obj2 = obj
                        does not create an independent copy of obj, but
                        only creates a handle to the same object, If you
                        change properties of obj, they will also show up in
                        obj2, and vice versa. the copy methods allows you
                        to create independent instances.
- AddRequests           spec.AddRequests(series,key,value1,[value2, ...])
                        adds values to an accumulating key.

The following methods have the same syntax:
- SaveRequests sets values to an accumulating key, overwriting existing
  values.
- KeepRequests removes all values not in the list of arguments.
- RemoveRequests removes all the values in the arguments list.
- RemoveKeys removes a whole key (or multiple keys) from a series.
- KeepKeys removes all keys except the ones in the arguments list.
- RemoveSections removes a whole series (or multiple series) from a spec.
- KeepSections removes all sections not in the arguments list.
- AddSections adds empty section(s).

spec.RemoveInconsistentSpecs cleans up spec so that it does not contain
  obvious inconsistencies. For instance, save-requests that only make
  sense for composites or for multiplicative decomopitions are removed if
  the conditions are not met. Many more cases are covered and cleaned
  up, although it is still possible that some inconsistencies remain.
spec.enforce(progname) changes a spec so that it is conformant with a
  particular program. For instance, spec.enforce('x12a.exe') makes the
  spec compatible with the X-12 version. spec.enforce('fixedseas.m')
  makes it compatible with fixedseas.m, etc.

spec.TransformPowerToFunction uses transform-power to set
  transform-function to either 'none' or 'log'.
spec.ExtractModeFormTransform uses transform-function to guess x11-mode
  (or custom-mode etc, depending on the adjustment method that is
  chosen).
ExtractValues extracts values from a spec and returns them in the correct
  format (as char, numerical, or cells).
disp displays a nicely formated content of the spec.
```

```
display is a short form of disp.
dispstring is the same as disp but returns a string that can be assigned
  to a variable.

Some static methods are also available:
[section,key] = legalize(section,key)
  This checks is a section is legal and is paired with a legal key
  belonging to that section. The input arguments can be abbreviated. The
  non-abbreviated versions are returned.
.toNum, .toCell, .toParen
  These methods transform any input (if possible) into a numeric, a cell
  or cellarray, and a char surrounded by parenthesis if there are
  multiple components, where the components are separated by spaces.
  (This is the way multiple values are stored in an .spc file).
```

x13series class

A x13series variable contains a seasonal adjustment of a single time series. It contains all the inputs, all the computed time series, and all the tables that are created in the process by the Census program. Like x13spec, the x13series has methods `display`, `disp`, and `dispstring`.

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
- 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 (removed, use .table('regression'))	string	result of regression as text (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)
- isempty	boolean	returns true if listofitems is empty
- 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.
- RunMfile            Runs an m-files that performs a seasonal
```

```
                      adjustment, 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.
```

table

```
  table returns the content of one or several tables contained in
  the object.

  Usage:
    Let obj be an x13series object [obj = x13(...)]. Then, ...
    obj.table returns a list of all tables
    obj.table('f') returns all tables whose heading starts with 'f'
        (i.e. 'f2' and 'f3')
    str = obj.table(...) places the output into a string variable
```

showvariable

```
showvariable returns a table with the content of a variable. Each column
is a period (month or quarter) and each row is one year.

Usage:
  showvariable(obj,name)
  showvariable(obj,name1,name2,...)
  tbl = showvariable(...)

Inputs:
  obj      A x13series of x13composite object.
  name     The three-letter name of a variable in obj.

Outputs:
  tbl      A table. If multiple names are given, tbl is a cell array of
           tables.

Example:

  load BoxJenkinsG;
  x = x13(BoxJenkinsG.dates,BoxJenkinsG.data,makespec('FULLX11'));
  x.showvariable('d9')

produces
                 Jan         Feb         Mar         Apr         May        ...
              _______     _______     _______     _______     _______

y1949             NaN         NaN         NaN         NaN         NaN        ...
y1950             NaN         NaN         NaN         NaN     0.92568        ...
y1951         0.91268         NaN      1.0544         NaN         NaN        ...
y1952         0.91579     0.94117         NaN         NaN         NaN        ...
y1953             NaN         NaN         NaN     0.98245     0.99152        ...
y1954             NaN     0.88632         NaN         NaN         NaN        ...
y1955             NaN         NaN         NaN         NaN         NaN        ...
y1956             NaN         NaN         NaN         NaN         NaN        ...
y1957             NaN         NaN         NaN         NaN         NaN        ...
y1958             NaN         NaN         NaN     0.94755         NaN        ...
y1959             NaN         NaN         NaN         NaN         NaN        ...
y1960             NaN     0.83732     0.96048     0.95257     0.98866        ...
```

plot

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

TO DO: legends and 'nolegend'
legends are not yet implemented, and 'nolegend' therefore has no effect yet.

MORE IMPORTANTLY, this code is a mess. It works, but it is extremely
difficult to maintain because it is not well structured. This requires a
fundamental overhaul...

Usage:
  plot(obj)
  plot(obj, 'variable1', 'variable2', ...)
  plot(obj1, obj2, ..., 'variable1', 'variable2', ...)
  plot(h, obj, ...)
  plot(..., 'columnwise'|'rowwise'|'layout',[rows,cols]|'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(..., 'fromdate',date)
  plot(..., 'todate',date)
  plot(..., 'selectdates',boolean vector)
  plot(..., 'nolegend')
  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.
  'layout',[rows,cols] sets the number of rows and columns of the
              subaxes. If rows or cols is NaN, it will be computed from
              the other one. It is not legal to set both to NaN.
  '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.
    'fromdate' or 'todate'  Boundaries of the dates that are represented on
                the hozizontal date axis in the graph.
    'selectdates'  This must be followed by a boolean vector that is as
                long as the time series that is being graphed. Only thos
                datapoints are represented in the graph who have a TRUE
                entry.
    'nolegend'  A legend is added when there is more than one variable that is
                printed. This option suppresses the legend.
    '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)');

To plot all variables in an x13series, try this:
    plot(x,x.listofitems{:});
```

seasbreaks

```
--- help for x13series/seasbreaks ---

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

  Usage:
    seasbreaks(obj)
    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 'd10' as well as 'd8'
     or 'd13' have been saved, or
   - the SEATS seasonal factors have been computed and 's10' and 's13' have
     been saved, or
   - any CUSTOM seasonal factors have been computed and 'sf' as well as 'si'
     or 'ir' have been saved.
  If SI (i.e. 'd8' or 's8') are missing, SI is recovered as SI = SF+IR (or
  (SI = SF*IR, depending on the type of adjustment defined in the spec).

  Inputs:
    obj          An x13series object.
    h            An optional figure handle.
    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.
```

x13toxls

x13toxls writes the content of 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.

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.
```

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.
```

addacf

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

NOTE: This method requires that Matlab's Econometrics Toolbox is
installed.

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.

NOTE: This method requires that Matlab's Econometrics Toolbox is
installed.

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');
```

addspectrum

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

NOTE: This method requires that Matlab's Signal Processing Toolbox is
installed.

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');
```

addMatlabSpectrum

```
addMatlabSpectrum computes the spectrum of a variable using the Signal
Processing. It adds up to four spectra (for Δdat, Δsa, Δir, and rsd). If
Spectra are already in the variable, they are replaced. Thus, this
program overwrites the spectra that were generated by x13as.exe.

NOTE: This method requires that Matlab's Signal Processing Toolbox is
installed.

Usage:
  x = x13(dates,data,spec);
  x.addMatlabSpectrum;
  plot(x,'sp0');
```

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 particular period(s) is removed from the data and the seasonal adjustment is computed again.

x13minus

```
x13minus takes two x13series objects and returns a new x13series object
that contains all time series both arguments have in common, but
containing the difference of the values. This is useful to exactly
compare the differences between two sets of specifications.

Usage:
  x3 = x13minus(x1,x2)

x1, x2, x3 are x13series objects. x3 contains all time series objects
that are common to x1 and x2, but with their differences as values. In
addition, x3 will contain variables 'tr','sa','sf','ir','si','rsd' that
contain the differences of the key variables, so that even if the key
variables have different names in x1 and x2, you can still get a
difference.

Example1:
  load BoxJenkinsG; dates = BoxJenkinsG.dates; data = BoxJenkinsG.data;
  spec1 = makespec('PICKFIRST','NOTRANS','EASTER','TD','X11', ...
      'series','name','linear');
  x1 = x13(dates,data,spec1);
  spec2 = makespec(spec1,'LOG','series','name','log');
  x2 = x13(dates,data,spec2);
  x3 = x13minus(x1,x2);
  ah = subplot(2,1,1); plot(ah,x1,x2,'e2','comb');
  ah = subplot(2,1,2); plot(ah,x3,'e2');

EXAMPLE 2:
  spec1 = makespec('PICKFIRST','LOG','EASTER','TD','X11', ...
      'series','name','X-11');
  x1 = x13(dates,data,spec1);
  spec2 = makespec(spec1,'SEATS','series','name','SEATS');
  x2 = x13(dates,data,spec2);
  x3 = x13minus(x1,x2);
  ah = subplot(2,1,1); plot(ah,x1,x2,'d11','s11','comb','quiet');
  ah = subplot(2,1,2); plot(ah,x3,'sa');
Note that x3.sa is x1.e2 - x2.s11, because the seasonally adjusted
variables have different names in in X11 and in SEATS.
```

Internally used method

```
PrepareFiles, Run, CollectFiles, RunMfile, runX12diag
addvariable, additem, rmitem, clean
updatemsg, showmsg, updatetables, addtable, rmtable
DetermineFrequency, Descrvariable
ExtractParagraph, ExtractSection, ExtractValue
```

Type `help x13series.Method` to receive short help on some method.

x13composite class

A composite seasonal adjustment is a procedure where a group of simultaneous time series are used to produce an alternative seasonal adjustment. In addition, the sum of these series is also computed and seasonally adjusted. The sum can be seasonally adjusted separately or can be indirectly adjusted by adding the seasonally adjusted series of its components. The Census program does both at the same time.

A x13composite variable contains a composite seasonal adjustment. It contains one x13series veriable for each components and an additional one for the aggregate.

Like x13spec and x13series, the x13composite has methods **display**, **disp**, and **dispstring**.

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).
```

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.
```

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.
```

Internally used methods

`PrepareFiles, Run, CollectFiles, RunOther, runX12diag`
`addseries, rmseries, ExtractParagraph`
`showmsg, updatemsg`

Type `help x13composite.Method` to receive short help on some method.

Seasonal adjustment without the Census program

The X-13ARMA-SEATS program from the Census Bureau does normally a very good job. But to use it, you need to copy the x13as.exe file from the Census website (which is done automatically with the `InstallMissingCensusProgram` command). This might not be possible for you, for instance because you work on a computer that is controlled by an IT administrator and you are not allowed to install any software. Moreover, the X-13 procedure is a bit of a black box, so we do not learn much about the mechanics of seasonal adjustment.

To address these issues, this toolbox provides alternatives that are completely done within Matlab. The toolbox contains four programs that perform a seasonal adjustment and do not use the Census programs:

- seas A simple, very transparent algorithm you can tinker with.
- fixedseas An algorithm that produces seasonal factors that do not change over time.
- x11 An approximate implementation of the older X-11 procedure that was developed by the Census bureau in the 1960s.
- camplet An algorithm that produces an adjustment that does not undergo revisions when new data are added to the time series.

Seas and fixedseas are variations of a common algorithm that contains of just four steps:

1. Use some form of smoothing to create a trend (TR) from the data (D).
2. Compute the deviation of D from TR (call it SI), make 12 series out of SI (for monthly observations, make 4 for quarterly observations, and likewise for other frequencies), and use again some smoothing algorithm on these 12 series separately. Call these smoothed series seasonal factors (SF).
3. Compute the difference between D and SF and call this the seasonally adjusted series (SA).
4. Compute the difference between TR and SA and call this the irregular series (IR).

We end up with a decomposition in the form S = TR + SF + IR, and SA = D – SF = TR + IR. TR contains the low frequency components of the data, SF contains the medium frequencies, and IR the high frequencies.

Step 1 is perfomed with the `trendfilter` function, Step 2 is performed with the `seasfilter` function.

To learn how to design your own seasonal adjustment algorithm, you can study the `seas.m` file, and, a bit more complicated, the `x11.m` and `fixedseas.m` files. The main choices you have is to use different methods to smooth, and to perform multiple rounds of such adjustments.

`x11.m` also just uses the basic components of trend-filters, seasonal filters, and normalization, but it is a fair bit more complicated than `seas.m`. This is an approximate implementation of the first X-11 algorithm and contains several stages of seasonal adjustment (based on movong averages).

`camplet.m` is quite a different filter. It relies only on backward looking filters in order to avoid revisions of the end of sample data when new data comes in.

seas

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

*** COMPONENTS OF THE seas SUBFOLDER **********************************

This folder contains a selection of programs that can be used to easily
create a seasonal adjustment, based on filters, yourself.

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_seas.m      Computes the difference or the ratio of two series,
                      depending on whether the decomposition is additive
                      or multiplicative.
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 program that produces a rather
                      simple version of seasonal adjustment in which the
                      seasonal factors are kept fixed over the years.
x11.m                 An implementation 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.
```

*** AN EXAMPLE: seas.M ***

Usage of seas.m

```
  s = seas(data,p)
  s = seas([dates,data],p)
  s = seas(...,[mode],[title])
```

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.

mode is either 'add', 'logadd', or 'mult'. 'add' implies that the seasonal factor will be zero on average and is subtracted from the unadjusted data to get to the seasonally adjusted data. If type is 'logadd', an additive decomposition is performed on the logarithm of the data, which are then converted back to their non-log versions afterwards. With 'mult', the seasonal factor is one on average, and the data is divided by the seasonal factor to get the seasonally adjusted data. Quantitatively, 'mult' should be quite simkilar to 'logadd'.

title is a string containing the name of the series (if one is provided).

s contains the output neatly organized in a struct. To make an x13series out of this, say the following,

```
  x = structtox13(s);
```

Alternatively, you can use the custom implementation with x13 as follows,

```
  x = x13(dates,data,spec,'prog','seas.m')
```

If you use it like this, the settings passed on are set in the spec in the 'custom' section, for instance,

```
  spec = makespec('custom','save','(sa sf)','custom','mode','add')
```

This version has the advantage that you can also specify trading day and Easter corrections, which are extracted via a regression of the irregular component of a first pass of seas.m, correctingh the data from that, and then running seas.m a second time.

*** MAKING YOUR OWN ***

You can easily make your own implementation. It may be easiest to start from seas.m and modify a copy of this file. Any custom m-file that performs a seasonal adjustment has to return a struct, containing, at the minimum, the following fields:

```
  'dates'     The column vector of dates.
  'dat'       The column vector of unadjusted data.
```

In addition, your output struct should contain the result of your

```
seasonal decomposition. Note that these fields must have names with at
most three letters (e.g., 'sa', 'rsd', etc). Fields with names longer
than three letters (except the ones listed below) will not be imported
into the x13series object.

Optional fields are:
  'keyv'      The content of this field is itself a struct with the
              following components: 'dat','tr','sa','sf','ir','si','rsd'.
              These fields contain the sames of key variables. This
              setting is stored not in the x13spec, but directly in the
              x13series object. If your output s does not contain a keyv
              field, the default is used,
              keyv = struct('dat','dat','tr','tr', 'sa','sa', ...
                            'sf','sf','ir','ir','si','si','rsd','rsd')
  'mode'      The mode of the adjustment (typically 'add', 'logadd',
              'mult', but others are possible, depending on what you
              implement). The setting is stored in custom-mode in the
              x13spec.
  'transform' A transformation of the data before processing (typically
              'none' or 'log', but again, more is possible. The setting
              is stored in transform-function in the x13spec.
  'title'     The title of the variable (if one is provided).
  'name'      The name of the series. There is a subtle difference
              between title and name. title can be any string, name
              should be a valid filename (this has to do with x13as.exe,
              which is irrelevant in this context, but it is good
              practice to observe this restriction anyway).
  'options'   Some content, to be defined by you, that describes any
              information or settings you wish to use in your seasonal
              adjustment. The setting is stored in custom-options in the
              x13spec.
  'tbl'       This is itself a struct. The content of this struct will be
              imported as tables into the x13series object.
```

x11

```
x11 computes an approximate version of the original X-11 seasonal
adjustment from 1965.

Literature: Dominique Ladiray et Benoît Quenneville, DÉSAISONNALISER AVEC
            LA MÉTHODE X-11, free version in French available for
            download from researchgate.net
            English version published as: Ladiray, Dominique, Quenneville,
            Benoit, "Seasonal Adjustment with the X-11 Method," Lecture
            Notes in Statistics, Springer, 2001.

CAUTION: The program computes only an approximate version of the original
X-11 algorithm. Most notably, data close to the edges of the sample are
treated differently, and there are some differences in the detection of
outliers.

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 occurs 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 in most cases.

An adjustment for calendar effects is not available using x11.m directly.
This is, however, implemented when using x11.m through x13.m as follows:
  x = x13(dates,data,spec);
If spec contains entries for 'regression','save','td' an adjustment for
trading days will be computed. Likewise, if spec contains
'regression','save','hol', an adjustment for Easter will be computed.
These corrections for calendar effects is different than the one
implemented in the original X-11. It also offers much less options than
the original, and also does not perform tests to determine whether
calendar adjustments are useful.

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 key steps of the
seasonal adjustment performed by the X-11 algorithm using Matlab directly. In
other words, this is a Matlab implementation of an approximate 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. Because the adjustment using X-11 is often quite similar
to the one offered by X-13, this program can be useful for users who are
unable to download or install the Census programs (for instance because
IT security regulation prevents installing executables).

Usage:
  s = x11(data,period);
  s = x11([dates,data],period);
  s = x11(... ,transform);
  s = x11(... ,transform,name);
  s = x11(... ,transform,name,dofull);

  s   This is a structure containing the following components:
      s.prog   = 'x11.m'
      s.name   = name of series (if given)
      s.period = period
      s.type   = type of decomposition
      s.tbl    = some calculations along the way
      s.dates  = dates vector
      s.dat    = data vector
      s.d10    = seasonal factor (cycle)
      s.d11    = seasonally adjusted data
      s.d12    = trend
      s.d13    = irregular component
                 .
                 .
                 .
      The other components are from intermediate computation steps. Their
      meaning is revealed in the documentation of x13as.exe.

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

name  is a string containing a descriptive title of the variable that is
      treated. This can be empty.

dofull
      is a boolean. If set to true, all the intermediate series are
      stored in the struct. Default is false, which means that only the
      most important final results are stored.

REMARK: This program uses several smaller programs (trendfilter, seasfilter,
normalize_seas) that can be used to create a custom seasonal adjustment algorithm
relatively easily. To understand how, just study the source code of this
program.
```

method1

```
method1 computes an approximate version of "Method I", developed by
Julius Shishkin in the 1950 at the US Census Bureau.

Note: I have not found a completely clear description of the algorithm,
so it is unlikely that the algorithm is exactly the same as the original.
My implementation is based on Allen H. Young's preface to the book by
Ladiray and Quenneville:
    "Dans la Méthode I, les coefficients saisonniers étaient estimés par
    l’intermédiaire de moyennes mobiles appliquées aux valeurs de la
    composante saisonnier-irrégulier de chaque mois. Cette composante
    saisonnier-irrégulier était elle-même calculée comme rapport de la
    série originale et du résultat du lissage de cette série originale
    par une moyenne mobile centrée sur 12 termes, lissage sensé
    représenter la composante tendance-cycle. Une seconde série ajustée
    était calculée, en remplaçant cette estimation de la tendance-cycle
    par le lissage de la première estimation de la série corrigée des
    variations saisonnières par une moyenne mobile simple d’ordre 5."
Also, the treatment at the edge of the sample is certainly different than
the original.

Usage:
  s = method1(data, period);
  s = method1([dates,data], period);
  s = method1(... , adjmode);
  s = method1(... , adjmode, title);

  s   This is a structure containing the following components:
        'prog',     'method1.m',  ...
        'tbl',      an explanatory text,  ...
        'title',    title,        ...
        'period',   period,       ...
        'mode',     adjmode,      ...
        'keyv',     struct('dat','dat','tr','d12','sa','d11','sf','d10', ...
            'ir','d13','si','d8','rsd','rsd'), ...
        'dates',    dates,        ...
        'dat',      data,         ...
        'd12',      tr,           ...
        'd8',       si,           ...
        'd10',      sf,           ...
        'd11',      sa,           ...
        'd13',      ir,           ...
        'b2',       tr1,          ...
        'b3',       si1,          ...
        'b4',       sf1,          ...
        'b6',       sa1);
```

```
adjmode
      must be one of the following: 'additive','none','multiplicative',
      or 'logadditive'. It indicates the type of decomposition.
      'additive' or 'none' : data = tr + sf + ir, sa = tr + ir.
          'multiplicative' : data = tr * sf * ir, sa = tr * ir.
             'logadditive' : log(data) = tr + sf + ir, sa = exp(tr + ir).

title is a string containing a descriptive title of the variable that is
      treated. This can be empty.
```

fixedseas

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

Usage:
  s = fixedseas(data,period);
  s = fixedseas([dates,data],period);
  s = fixedseas(... ,mode);
  s = fixedseas(... ,smoothmethod);
  s = fixedseas(... ,smoothmethod,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.

'mode' is one of the following:
  'none' or 'add'     The decomposition is done additively. This
                      is the default.
  'logadd             The log is applied to the data, the
                      decomposition is then applied additively,
                      and the exponential of the result is
                      returned.
  'mult'              The decomposition is done multiplicatively.

'smoothmethod' (and 'methodarg') determines the method of trend. There
are many choices here, see ''help trendfilter'' for a description.
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, 'mode' and 'smoothmethod' 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 mode 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.
```

```
Depending on the method used, 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.
  .mode        Either 'none' or 'log' or 'mult'.
  .smoothmethod  The method used for computing the trend.
  .methodarg   possibly a parameter for the smoothing algorithm.
  .tbl         A short explanation of the algorith.
  .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, polynomial, or others (see help trendfilter).]
*** 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 average' 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.

NOTE: This file is part of the X-13 toolbox, but it is completely
independent of the Census X-13 program. It is part of the 'seas' addition
to the toolbox which allows to implement seasonal filters without using
the Census Bureau programs.
```

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;
```

NOTE: This program is part of the X-13 toolbox, but it is completely independent of the Census X-13 program. It uses a simpler strategy to filter seasonal cycles than X-13ARIMA-SEATS. The main advantage of camplet is that this argorithm does not produce revisions of older seasonal adjustements when new data comes in. Also, camplet accomodates arbitrary frequencies, not only monthly and quarterly. Moreover, the residual seasonality is often much smaller than when using fixedseas.m, but unlike with this algorithm, the seasonal factors are not constant, but adapt over time. This program is just a small addition to the toolbox that makes it more complete.

spr

```
spr computes the standard deviation of the irregular from the seasonal
filtering done with fixedseas, seas, or x11, 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, data2, ...);
  [s,p,r,x] = spr(..., method);
  [s,p,r,x] = spr(..., options);

data or data1, data2, etc are individual vectors or data.

method is one of the following: 'fixedseas',' seas', 'x11'. Default is
'fixedseas'.

options are any options passed on to fixedseas/seas/x11.

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,'add');
  % The graph reveals a clear spike at 14 (and an echo at 28 etc), which
  % we filter out now ...
  s = fixedseas(data,14,'add');
  figure; spr(s.sa,s.ir,'add');
  % 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],'add');
  figure; spr(s(end).sa,s(end).ir,'add');
  % The seasonally adjusted series and the residuals show no spikes
  % anymore.
```

NOTE: This program is part of the X-13 toolbox, but it is completely independent of the Census X-13 program. It uses fixedseas to filter seasonal cycles and computes the volatility of the resulting residuals. This program is just a small addition to the toolbox that makes it more complete.

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)

NOTE: This program is part of the X-13 toolbox, but it is completely
independent of the Census X-13 program. It is part of the 'seas' addition to
the toolbox which allows to implement seasonal filters without using the
Census Bureau programs.
```

normalize_seas

```
normalize_seas computes the additive or multiplicative difference of two time
series

Usage:
  adj = normalize_seas(data,trend,[ismult])

data and trend must be a vectors of equal length. ismult is a boolean.

adj ist either data-trens (is ismult is missing or false), and
data./trend is ismult is true.

NOTE: This program is part of the X-13 toolbox, but it is completely
independent of the Census X-13 program. It is part of the 'seas' addition to
the toolbox which allows to implement seasonal filters without using the
Census Bureau programs.
```

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.
  '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.
In addition, all the kernels supported by kernelweights.m can also be
specified here:
  '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).
One of the following:
  'uniform','triangle','biweight' or 'quartic','triweight','tricube',
  'epanechnikov', 'cosine','optcosine','cauchy', followed by a single
   parameter indicating the bandwidth.
Some kernels have infinite support: 'logistic','sigmoid','gaussian' or
  'normal','exponential','silverman'. If you choose one of these, you can use
  two parameters, the first indicating the bandwidth, the second indicating
  the length of the vector that is returned. (If the second parameter is not
  given, a vector is returned where all elements are at least 1e-15.

'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'.

NOTE: This program is part of the X-13 toolbox, but it is completely
independent of the Census X-13 program. It is part of the 'seas' addition to
the toolbox which allows to implement seasonal filters without using the
Census Bureau programs.
```

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;

NOTE: This program is part of the X-13 toolbox, but it is completely
independent of the Census X-13 program. It is part of the 'seas' addition to
the toolbox which allows to implement seasonal filters without using the
Census Bureau programs.
```

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; d = unemp.data; n = numel(d);
  m = splitperiods(d,12);
```

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

```
  nanmean(m)

  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.

We can also plot the data month-wise

```
  plot(m,'linewidth',1); grid on;
  legend('Jan','Feb','Mar','Apr','May','Jun','Jul','Aug','Sep','Oct', ...
      'Nov','Dec');
```

NOTE: This program is part of the X-13 toolbox, but it is completely independent of the Census X-13 program. It is part of the 'seas' addition to the toolbox which allows to implement seasonal filters without using the Census Bureau programs.

joinperiods

```
joinperiods is the opposite of splitperiods.

Example:
  data = [100 90 85 150 101 90 86 155 101 88 87 152 98 91 84 144 ...
      99 88 86 153]';
  q = splitperiods(data,4); nyears = size(q,1);
  plot(q); title('data by quarter');
  legend('First','Second','Third','Fourth Quarter');
  qadj = q - repmat(mean(q),nyears,1) + mean(data);
  dataadj = joinperiods(qadj);
  figure; plot([data,dataadj],'linewidth',1); grid on;
  title('unadjusted and adjusted data');

NOTE: This program is part of the X-13 toolbox, but it is completely
independent of the Census X-13 program. It is part of the 'seas' addition to
the toolbox which allows to implement seasonal filters without using the
Census Bureau programs.
```

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 moving 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 polynomial of order n,
                               and minimizes 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. If 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 1:
m = kernelweights('ma',5,5,4,4);
h = kernelweights('henderson',15);
b = kernelweights('bongard',15);
s = kernelweights('spencer');
```

```
e = [0;kernelweights('epanechnikov',15);0];
n = kernelweights('normal',2,15);
plot((-7:7),[m,s,h,b,e,n],'linewidth',1);
legend('4-fold MA','Spencer','Henderson(15)','Bongard(15)', ...
   'Epanechnikov(15)','Gaussian(2)');
xlim([-7,7]);
grid on;

Example 2:
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;

NOTE: This program is part of the X-13 toolbox, but it is completely
independent of the Census X-13 program. It is part of the 'seas' addition to
the toolbox which allows to implement seasonal filters without using the
Census Bureau programs.
```

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,[direction])

  data    An array of data, organized columnwise.
  w       a vector with an odd number of values.
  direction is one of 'centered', 'backwards', or 'forward'. Default is
          'centered'.
  s       s is the convolution of data and w.

If direction is set to 'backwards' ('forward'), all the weigths in the
right (left) half of w are set to zero, leading to a non-centerd weighted
average.

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])]);

NOTE: This program is part of the X-13 toolbox, but it is completely
independent of the Census X-13 program. It is part of the 'seas' addition to
the toolbox which allows to implement seasonal filters without using the
Census Bureau programs.
```

Tools for working with dates

The toolbox contains two programs

- `makedates`
- `yqmd`

that make it easier to generate or work with date vectors.

In addition, `TakeDayOff` is a procedure that allows you to identify dates when it is likely that many people would take a day off from work, whoich can be an important exogenous variable to predict calendar day effects.

makedates

```
makedates returns a vector of dates with given frequency.

Usage:
  d = makedates(startdate,enddate,period,[mult])

startdate and enddate are calendar days indicating when the series should
start and end. They can be given as simple three-component vectors
[year,month,day], as datenum numbers, or as datetime variables.

period is one of the following: 'year', 'semester', 'trimester',
'quarter', 'month', 'week', 'weekday', or 'day'.

mult is a multiple: using period='day' and mult=3 returns every third
day. 'day',7 is equivalent to 'week',1, 'month',3 is equivalent to
'quarter',1, etc. However, 'week',4 is not equivalent to 'month',1
because a month does not contain exactly four weeks. mult must be a
positive integer. Default for mult is 1.

Example 1:
    d = makedates([2019,2,1],[2020,5,20],'quarter');
    disp(datestr(d));
    01-Feb-2019
    01-May-2019
    01-Aug-2019
    01-Nov-2019
    01-Feb-2020
    01-May-2020
Example 2:
    d = makedates([2019,1,31],[2019,7,15],'month');
    disp(datestr(d));
    31-Jan-2019
    28-Feb-2019
    31-Mar-2019
    30-Apr-2019
    31-May-2019
    30-Jun-2019
    Note: If the day in the startdate is the last day of the month, then
    the last day of the month will be used for all entries.
Example 3:
    d = makedates([2019,1,1],[2019,3,31],'week',2);
    disp(datestr(d));
    01-Jan-2019
    15-Jan-2019
    29-Jan-2019
    12-Feb-2019
    26-Feb-2019
    12-Mar-2019
    26-Mar-2019

NOTE: This program is part of the X-13 toolbox, but it is completely
independent of the Census X-13 program and can be used even if the census
programs are not installed.
```

yqmd

```
yqmd retrievs the year, quarter, month, or day of a (vector of) serial dates.
This is used for versions of Matlab prior to R2013a, before the 'year',
'quarter', 'month', and 'day' commands were introduced in Matlab.

Usage: d = yqmd(d,type)

type must be one of the following: 'year', 'semester', 'trimester',
'quarter', 'month' or 'm', 'day', 'weekday', 'hour', 'minute', 'second', or
abbreviations thereof.
```

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);
```

EasterDate

```
EasterDate computes date of Easter for the western and the orthodox churches

Usage:
  [d,g,j] = EasterDate(y,['western' or 'eastern' or 'orthodox']);

y      A year or a vector of years.
d    The date of Easter as a datenum.
g      A [year,month,day] vector containing the date in the Gregorian
       calendar.
j      The same as g but using the Julian calendar. This is only returned
       if the orthodox version of the Easter date is computed.
'w','e','o'   Indicates if the western or the orthodox (eastern) Easter
       date is to be computed. 'eastern' is synonymous to 'orthodox'.

Example:
w = EasterDate(2020:2030);
o = EasterDate(2020:2030,'orth');
[datestr(w),repmat('; ',11,1),datestr(o)]
    '12-Apr-2020; 19-Apr-2020'
    '04-Apr-2021; 02-May-2021'
    '17-Apr-2022; 24-Apr-2022'
    '09-Apr-2023; 16-Apr-2023'
    '31-Mar-2024; 05-May-2024'
    '20-Apr-2025; 20-Apr-2025'
    '05-Apr-2026; 12-Apr-2026'
    '28-Mar-2027; 02-May-2027'
    '16-Apr-2028; 16-Apr-2028'
    '01-Apr-2029; 08-Apr-2029'
    '21-Apr-2030; 28-Apr-2030'

The program implements the Meeus/Jones/Butcher algorithm for the Western
Easter date and the Meeus algorithm for the Orthodox Easter date, see
https://en.wikipedia.org/wiki/Computus
```

Casting

A class in Matlab is really a structured variable (a struct), embellished with class-specific functions. It is therefore natural to allow the user to cast an ordinary struct into a x13series.

structtox13

```
structtox13 uses the content of a struct to create a x13object

Usage:
  x = structtox13(s)

s is a struct. It must at least contain the fields 'dates', 'dat',
'period', and either 'type' or 'transform'.

  .period   is the number of observations (positive integer) for the
            cycle that is removed, 'type' or 'transform' is either
            'additive', 'multiplicative', 'logadditive' or some other
            transform function supported by the program you use,
            indicating the type of decomposition.
 .dates     These are column vectors containing the dates of observation
 .dat       (datenum) and the observations (floats). Both must have the
            equal length. x is a x13series object containing the same
            information.

Optional fields that are treaded in a special way are 'name' and 'prog'.
 .name      is the name of the variable that is seasonally adjusted (if
            this content is a string).
 .prog      is the name of the program that was used to perform the
            seasonal adjustment.

All other fields are added to the x13series object if their fieldnames
have at most three characters. (Variablenames in x13series objects are
constrained to three-letter names).

structtox13 sets up the x13series variable and adds the dates and data
vector to it, and then defers to addstructtox13 to do the rest of the
work.
```

addstructtox13

```
addstructtox13 uses the content of a struct and adds it to a x13object

Usage:
  x = structtox13(x,s)

s is a struct. It must at least contain the fields 'dates', 'dat',
'period', and either 'type' or 'transform'.

  .period   is the number of observations (positive integer) for the
            cycle that is removed, 'type' or 'transform' is either
            'additive', 'multiplicative', 'logadditive' or some other
            transform function supported by the program you use,
            indicating the type of decomposition.
 .dates     These are column vectors containing the dates of observation
 .dat       (datenum) and the observations (floats). Both must have the
            equal length. x is a x13series object containing the same
            information.

Optional fields that are treaded in a special way are 'name' and 'prog'.
 .name      is the name of the variable that is seasonally adjusted (if
            this content is a string).
 .prog      is the name of the program that was used to perform the
            seasonal adjustment.

All other fields are added to the x13series object if their fieldnames
have at most three characters (variable names in x13series objects are
constrained to three-letter names).
```

Backward Compatibility Issues

Version 1.50 of the toolbox has seen considerable changes over previous versions. As a result, it was not possible (or it would have been very cumbersome) to uphold all aspects of the syntax to ensure compatibility with previous versions of the toolbox. If you have written a script that uses the X-13 Toolbox, you therefore might have to adapt it.

Only *one* algorithm per x13series instance

In earlier version, it was possible to have, say, X11 and FixedSeas adjustments in the same x13series variable. This is no longer the case. This was limited anyway. For instance, it was not possible to mix X11 with SEATS. For clarity, the mixture of algorithms in one instance has been removed. You can still work with multiple algorithms on the same data, of course, You only need to have separate variables containing the results.

To generate an adjustment with the X-11 program or using the approximate X-11 algorithm (as implemented in x11.m), one says, for instance,

```
x = x13(dates,data,makespec('X11', 'LOG', 'EASTER'), 'x-12')
x = x13(dates,data,makespec('X11', 'LOG', 'EASTER'), 'x-11')
```

In previous versions of the toolbox, the syntax was differnt for using 'fixedseas' or 'camplet',

```
x = x13(dates,data,makespec('FIXED', 'LOG', 'EASTER'))
x = x13(dates,data,makespec('CAMPLET', 'LOG', 'EASTER'))
```

generated an adjustment using the fixedseas or camplet algorithms, respectively.

This does no longer work. Now it is necessary to provide 'camplet' as an argument to x13, instead of just having it as a section in the spec.

```
x = x13(dates,data,makespec('FIXED', 'LOG', 'EASTER'),'fixedseas')
x = x13(dates,data,makespec('CAMPLET', 'LOG', 'EASTER'),'camplet')
```

This solution improves consistency of the syntax across algorithms.

seas.m

I have taken the liberty already in pervious iterations of the toolbox to freely change `seas.m`. This m-file is intended as an example how to implement a custom seasonal adjustment algorithm. You are supposed to play with it. The current version is different than the previous ones, so if you relied on the previous version, just give the new `seas.m` another name, so that you can keep using the previous implementation.

x11.m and method1.m

`x11.m` has also been completely re-written. It is hardly recognizable compared to the previous version. I believe that it is now much closer to the original X-11 as developed in 1965 by Shishkin and others.

There is also a new `method1.m` seasonal adjustment algorithm that is likely quite close to the Census Bureau Method I that was used even before that. If you are interested how it works you can study the code. This is surprisingly straightforward to implement using the tools of the seas subdirectory. It is just nine lines of code:

```
tr1 = trendfilter(data,'cma',p,'mirror',ceil(p/2));
si1 = normalize_seas(data,tr1,ismult);
sf1 = seasfilter(si1,p,'ma',[3,3],'mirror',3);
sa1 = normalize_seas(data,sf1,ismult);
tr  = trendfilter(sa1,'ma',5,'mirror',3);
si  = normalize_seas(data,tr,ismult);
sf  = seasfilter(si,p,'ma',[3,3],'mirror',3);
sa  = normalize_seas(data,sf,ismult);
ir  = normalize_seas(sa,tr,ismult);
```

calendar adjustments

A simplified version, based on regressions on the irregular component, has been implemented. This means that if `td` or `hol` are saved in the specification, a calendar adjustment will be performed also for the custom seasonal adjustment algorithms, i.e. `x11`, `method1`, `seas`, `fixedseas`. This does not work with `camplet` because this algorithm misses an irregular component. Also, the new method has no effect when using the original Census Bureau exe files (i.e. x-12 or x-13).

The method works as follow: A first pass of the seasonal adjustment is computed. The resulting irregular component (containing only high frequency fluctuations) is then regressed on dummies that count the number of different types of weekdays in a month, and the location of Easter in the calendar. The original data are then adjusted for the variations that are explained by the calendar dummies, and the seasonal adjustment is run again on these adjusted data. This procedure is inspired by the original's regARIMA, and is relatively effective in removing trading-day frequency spikes in the spectra.

add, logadd, mult

Some of the custom seasonal adjustments allowed a long form to be specified for the adjustment mode (`'additive'`,`'logadditive'`,`'multiplicative'`). This has been changed, and only the shorter forms are supported now.

Also, makespec has two new collective sets, LOG and NOTRANSFORM. LOG is equivalent to MULT, and NOTRANSFORM is equivalent to ADD. This has been done to enhance clarity. After all, MULT does not really induce a multiplicative adjustment, but a log-additive one.

To shed some light on this issue: the SEATS and X11 algorithm allows the user to apply a variety of transformations of the data before processing them. This is relevant for the regARIMA stage. By taking the log of the data, you get effectively a multiplicative decomposition, or more precisely, a log-additive decomposition

    log(data) = log(tr) + log(sf) + log(ir).

although these terms are not used at least by SEATS.

X11 also has a 'mode' key is set in the 'x11' section, which accepts one of four values: 'add', 'logadd', 'pseudeadd', or 'mult'. This is the only way to achieve a truly multiplicative decomposition

    data = tr * sf * ir.

Because the 'mode' key is only available when using the X11 algorithm, there is no way to force a truly multiplicative decomposition in all cases. For this reason, in `makespec`, 'LOG' is a more precise description of what is done than 'MULT' (which is kept only for backward compatibility). Likewise, 'NOTRANSFORM' is a more precise description of the fact that the data are not transformed before treatment, but does not by itself imply an additive decomposition, hence 'ADD' is misleading.

Here’s an example

```
      spec = makespec('X11','MULT')
```

is the same as

```
      spec = makespec('X11','LOG')
```

This just adds the entries `'transform','function','log'`, but leaves `'x11','mode'` unaffected.

To force a truly multiplicative decomposition you can say

```
      spec = makespec('X11','x11','mode','mult')
```

The multiplicative decomposition is supported by X11 (so x-13, x-12, also the approximate x-11 and Method I), as well as by FIXED and SEAS. It is not supported by CAMPLET. Forcing a multiplicative decomposition with a custom algorithm (using an m-file instead of a Census exe), the syntax is analogous, as shown here as an example,

```
      spec = makespec('FIXED','fixed','mode','mult')
```

The custom m-files for seasonal adjustment do not support regARIMA (except for calendar adjustment), but do support three of the X11 modes, `'add'`, `'logadd'`, `'mult'`. The `'logadd'` setting simply takes the log of the data first, then performs an additive decomposition, and un-logs the result in the end. `'pseudoadd'` is not supported, and other transformations of the data are also not supported (but you can transform the data in Matlab before processing them with the toolbox in any way you like, of course).

A new 'custom' section in x13spec

X13spec now features a new section called 'custom'. This section accepts the keys 'period', 'mode', 'adjtype', 'options', and 'save'. The contents of this section are used to determine what to do when using a custom m-file for seasonal adjustment (such as seas.m). Any variables that should be saved need to be listed in custom-save.

Likewise, when using fixedseas, the variables that should be saved must be listed in fixedseas-save. The same logic applies to camplet.

This is not true for x11 or method1. These two algorithms are governed by the x11 section.

Some properties have been removed from x13series, some have been added to x13spec

The .title, .name, and .filename properties of the x13series class have been removed. Instead, x13spec now has a property .title and .name. These are derived from the content of the spec. x13 reads the .name property specified by the user in the spec. If that is missing, it derives a valid file name from the .title property, and writes it into the spec. For this reason, a separate .filename property is no longer required.

The .isLog property has also been removed from x13series. Instead, x13spec has gained a few properties. .transfunc returns the function used to transform the data before treatment, .adjmode returns the type of decomposition that is used.

There is also another new property called .adjmethod in x13spec that returns the name of the seasonal adjustment algorithm that is used (x11, seats, camplet, ...).

x13series.keyv

The x13series class now contains a new property called .keyv. This property contains a struct with the fields 'dat', 'tr', 'sa', 'sf', 'ir', 'rsd'. These fields contain the names of the most important variables, which do depend on the algorithm used. For instance, the seasonally adjusted series is called 'd11' in X11, but is called 's11' in SEATS.

If you design your own seasonal adjustment m-file, make sure to incluse a corresponding keyv field in your struct.

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);
size10 = sizeFig(70,80);

% 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 travel
report(['Source and discription of data: ',travel.source,newline]);
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.',newline]);

spec1a = makespec('AUTO','TRAMOPURE','DIAG','series','name',travel.descr);
travel1a = x13(travel.dates,travel.data,spec1a,'quiet');

disp(travel1a.table('transform'));
report(['CONCLUSION: The filtering will be additive.',newline]);

spec1 = makespec(spec1a, 'NOTRANS');
travel1 = x13(travel.dates,travel.data,spec1,'quiet');

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

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

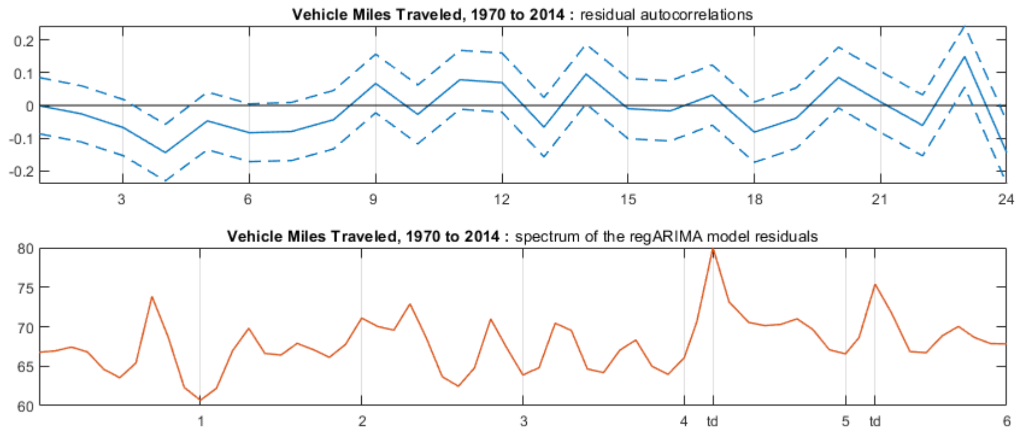
```
------------------------------------------------------------------

  *****   AICC (with aicdiff=-2.00) prefers no transformation   *****

CONCLUSION: The filtering will be additive.

CONCLUSION: The data are clearly seasonal.
```

Step 2: Calendar Dummies

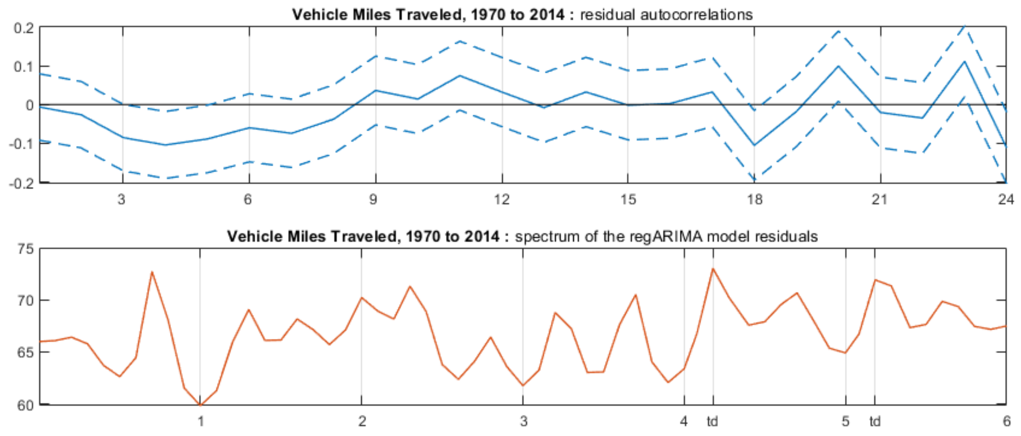
```
disp(sline)
fprintf(' Step 2: Optimizing the regression\n\n');

report(sprintf(['TRAMO has chosen an ARIMA %s, but the autocorrelation ', ...
    'function is problematic. We have significant autocorrelation of the ', ...
    'residuals.\n'], travel1.arima));

fh = figure('Position',size2);
plot(fh,travel1,'acf','spr');

report(sprintf(['The problem could be the lack of a dummy for Easter and ', ...
    'trading days. It is very likely that Easter as well as the distribution ', ...
    'of weekends plays a role in travel behavior. We therefore add ', ...
    'Easter and trading day dummies and check if that solves the problem.\n']));

spec2a = makespec(spec1,'EASTER','TD');
travel2a = x13(travel.dates,travel.data,spec2a,'quiet');

disp(travel2a.table('regression'));
disp(travel2a.table('tukey'));

fh = figure('Position',size2);
plot(fh,travel2a,'acf','spr');

report(sprintf(['\nThe trading days are significant, but Easter is not. The ', ...
    'algorithm kicks out this dummy when it is not significant enough. We will ', ...
    'therefore keep the Easter dummy in for the moment; maybe it will become ', ...
    'relevant later. Also, the trading day dummies have not yet resolved the ', ...
    'autocorrelation issue.\n'], ...
    travel2a.arima));

report([newline,'The Tukey report shows a problem at frequency 6 in the ', ...
    'but a visual inspection of the spectrum also indicates a problem at ', ...
    'trading day frequencies. We try to address this by also adding ', ...
    'dummies for labor day and thanksgiving. This does not fully solve ', ...
    'the problem, however.',newline]);

ti = '(labor[1] thank[1])';
s11 = makespec(spec2a, 'FORCETD', 'regression','variables',ti, 'series','name',ti);
ti = '(labor[1] thank[8])';
s18 = makespec(spec2a, 'FORCETD', 'regression','variables',ti, 'series','name',ti);
ti = '(labor[8] thank[1])';
s81 = makespec(spec2a, 'FORCETD', 'regression','variables',ti, 'series','name',ti);
ti = '(labor[8] thank[8])';
s88 = makespec(spec2a, 'FORCETD', 'regression','variables',ti, 'series','name',ti);

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');

fh = figure('Position',size8);
```

```
plot(fh,t11,t18,t81,t88,'acf','spr');

report(['We have tried combinations of thank[1] and thank [8] on the one hand ', ...
    'and labor[1] and labor[8] on the other. All of these specifications are ', ...
    'very similar. The best likelihood is achieved with the labor[8] and ', ...
    'thank[1] combination, so we will include these dummies.',newline]);

spec2 = makespec(s81,'series','name',travel.descr);
travel2 = t81;
```

```
--------------------------------------------------------------------------------
 Step 2: Optimizing the regression

 TRAMO has chosen an ARIMA (2 1 0)(0 1 1), but the autocorrelation function is
 problematic. We have significant autocorrelation of the residuals.
```



```
The problem could be the lack of a dummy for Easter and trading days. It is
 very likely that Easter as well as the distribution of weekends plays a role
 in travel behavior. We therefore add Easter and trading day dummies and check
 if that solves the problem.
```



```
Estimation converged in   10 ARMA iterations,   89 function evaluations.

 Regression Model
```

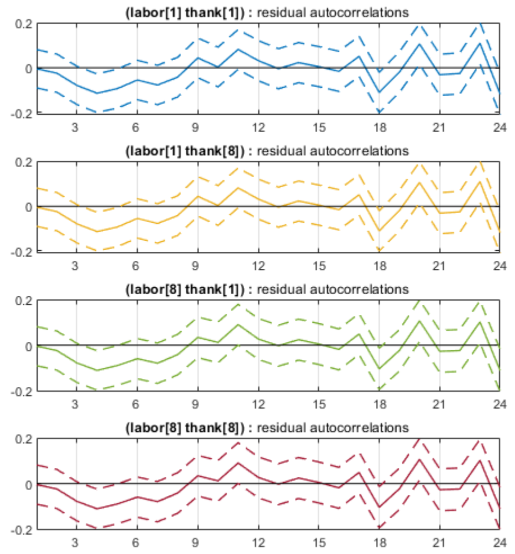
```
-------------------------------------------------------------------------------
                            Parameter        Standard
Variable                     Estimate           Error      t-value
-------------------------------------------------------------------------------
Leap Year                   1043.8270       638.13205         1.64

Trading Day
  Mon                        -86.4250       200.08303        -0.43
  Tue                        129.4466       198.94503         0.65
  Wed                        -51.7153       199.65579        -0.26
  Thu                        644.7656       201.06038         3.21
  Fri                        478.4668       200.41043         2.39
  Sat                       -588.8076       200.04820        -2.94
 *Sun (derived)             -525.7312       199.89926        -2.63
-------------------------------------------------------------------------------
 *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           86.83         0.00
Combined Trading Day and Leap Year Regressors
                                     7           89.56         0.00
-------------------------------------------------------------------------------
Trading Day                          6, 518          14.28         0.00

ARIMA Model:  (2 1 0)(0 1 1)
  Nonseasonal differences: 1
  Seasonal differences:    1
                                             Standard
Parameter                    Estimate          Errors
---------------------------------------------------
Nonseasonal AR
  Lag  1                      -0.4389         0.04293
  Lag  2                      -0.1756         0.04276

Seasonal MA
  Lag 12                       0.6628         0.03307

Variance                       0.66997E+07
SE of Var                      0.41352E+06
---------------------------------------------------

Likelihood Statistics
------------------------------------------------------------------
Number of observations (nobs)                                  538
Effective number of observations (nefobs)                      525
Number of parameters estimated (np)                             11
Log likelihood (L)                                      -4874.3855
AIC                                                      9770.7710
AICC (F-corrected-AIC)                                   9771.2857
Hannan Quinn                                             9789.1349
BIC                                                      9817.6684
```

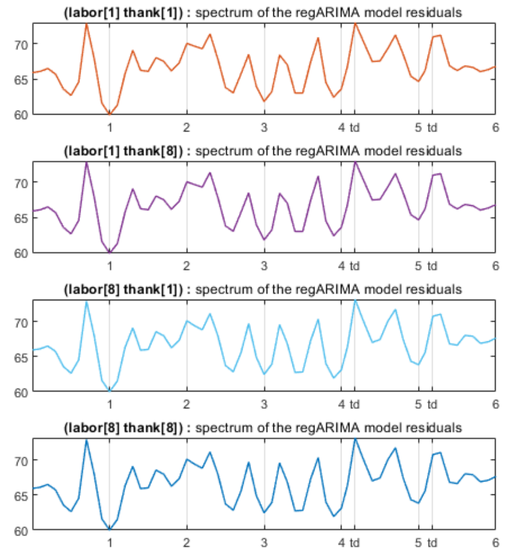
```
-----------------------------------------------------------------

 Roots of ARIMA Model
  Root                    Real  Imaginary    Modulus  Frequency
  -----------------------------------------------------------
  Nonseasonal AR
   Root  1             -1.2493     2.0328     2.3860     0.3377
   Root  2             -1.2493    -2.0328     2.3860    -0.3377
  Seasonal MA
   Root  1              1.5087     0.0000     1.5087     0.0000
  -----------------------------------------------------------

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

                                              S1       S2       S3       S4       S5       S6       TD
                                            ------   ------   ------   ------   ------   ------   ------
  Model Residuals                            0.561    0.293    0.930*   0.005    0.270    0.914*   0.527
  Prior Adjusted Series (Table B1)           0.960*   0.999**  0.998**  0.990*   1.000**  0.785    0.031
  ----------
     ** - Peak Probability > 0.99,
      * - 0.90 < Peak Probability < 0.99
```

The trading days are significant, but Easter is not. The algorithm kicks out this dummy when it is not significant enough. We will therefore keep the Easter dummy in for the moment; maybe it will become relevant later. Also, the trading day dummies have not yet resolved the autocorrelation issue.

The Tukey report shows a problem at frequency 6 in the but a visual inspection of the spectrum also indicates a problem at trading day frequencies. We try to address this by also adding dummies for labor day and thanksgiving. This does not fully solve the problem, however.

We have tried combinations of thank[1] and thank [8] on the one hand and labor[1] and labor[8] on the other. All of these specifications are very similar. The best likelihood is achieved with the labor[8] and thank[1] combination, so we will include these dummies.

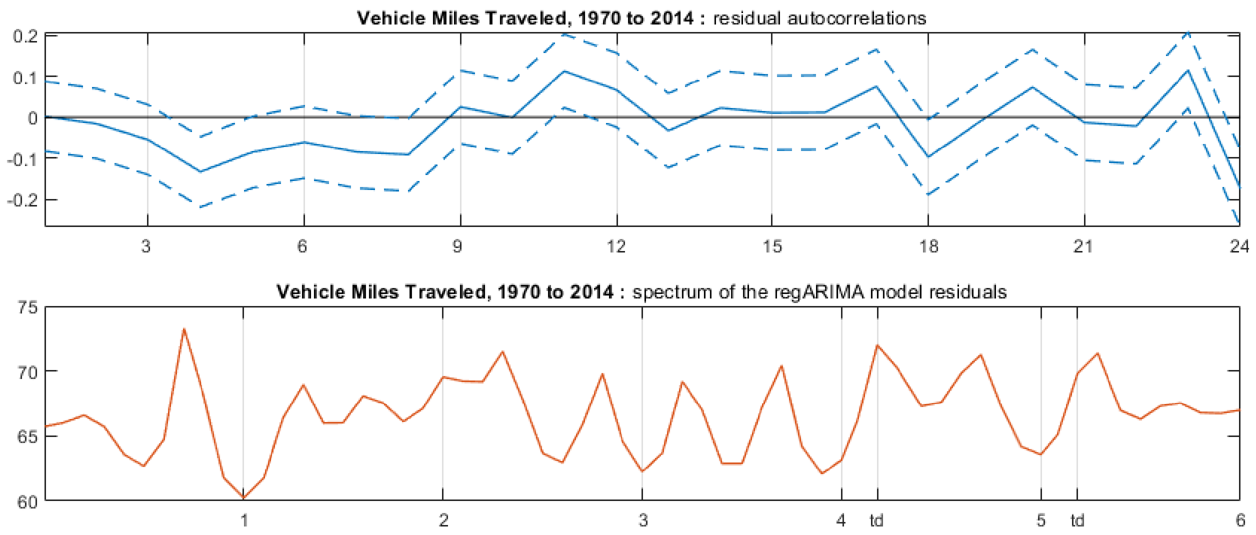




Step 3: Automatic detection of structural breaks and outliers

```
disp(sline)
fprintf(' Step 3: Finding structural breaks\n\n');

report(['Next we allow the algorithm to detect one-time outliers and ', ...
    'level shifts, ba adding the AO and LS options.',newline]);

spec3 = makespec(spec2,'AO','LS');
travel3 = x13(travel.dates,travel.data,spec3,'quiet');

fh = figure('Position',size2);
plot(fh,travel3,'acf','spr')

report(['Two ouliers have been detected, a level shift LS1979.May and a ', ...
    'one-time outlier AO1995.Jan.',newline,newline,'The spikes in the ', ...
    'spectrum are now under control, but the autocorrelation issue remains.', ...
    newline]);
```

```
-------------------------------------------------------------------------------
 Step 3: Finding structural breaks

 Next we allow the algorithm to detect one-time outliers and level shifts, ba
 adding the AO and LS options.

 Two ouliers have been detected, a level shift LS1979.May and a one-time
 outlier AO1995.Jan.

 The spikes in the spectrum are now under control, but the autocorrelation
 issue remains.
```

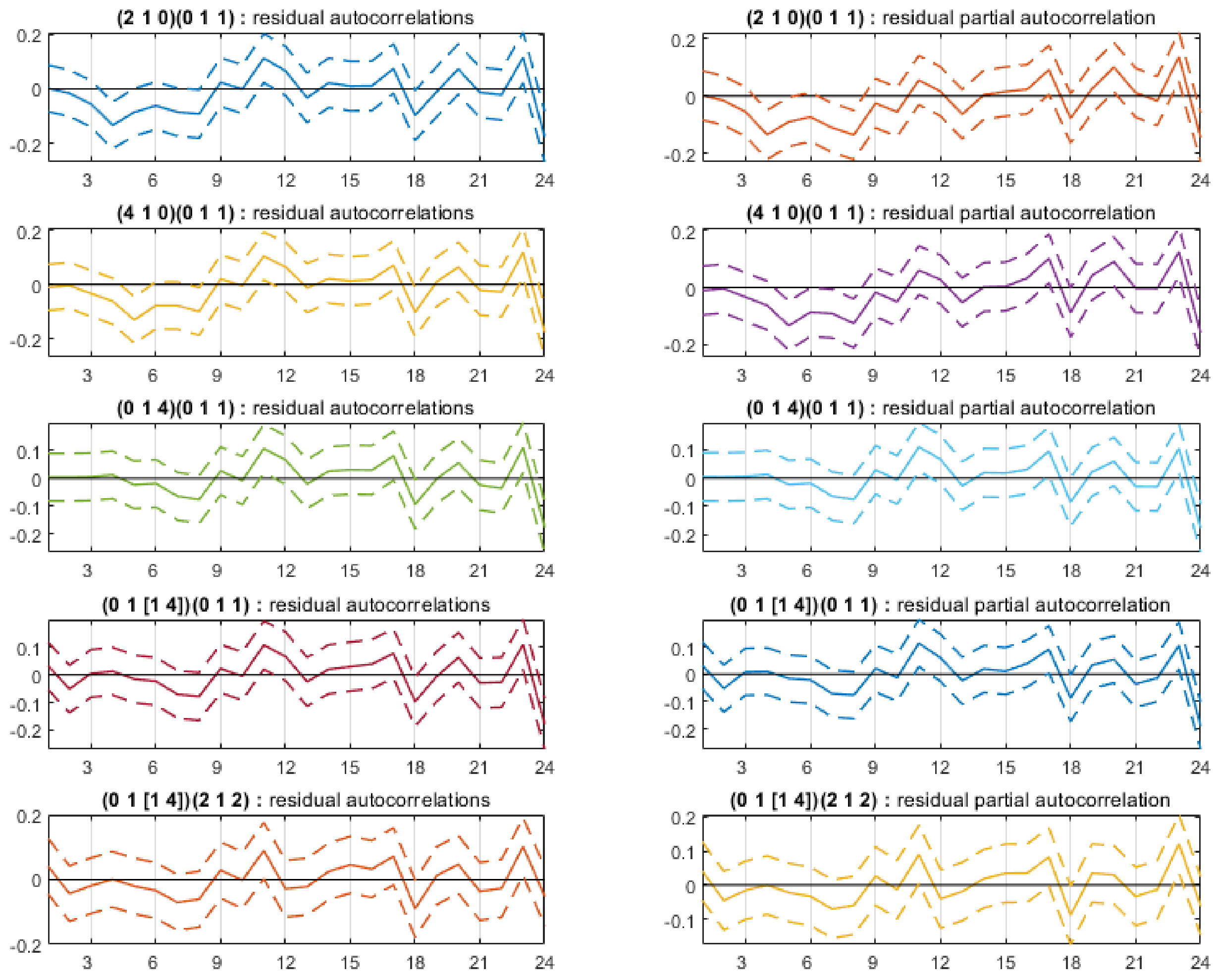


Step 4: Tweaking the ARIMA

```
disp(sline)
fprintf(' Step 4: Tweaking the ARIMA\n\n');

report(['The Spectrum and autocorrelation problems have still not been ', ...
    'resolved. It seems that we have to tweak the ARIMA specification ', ...
    'manually. Inspecting the ACF, there is a significant problem at lag ', ...
    '4 and maybe at lag 18. Such a long lag (18) would normally not be an ', ...
    'issue, but it might indicate an inappropriate choice of the seasonal ', ...
    'part of the ARIMA, since 18 lags is 1.5 years.', newline,' ',newline, ...
    'But we address the problem at lag 4 first. We check (4 1 0) and (0 1 4) ', ...
    'and find that (0 1 4) works much better.', newline,' ',newline, ...
    'However, looking at the regression output, we notice that the 2nd and 3rd ', ...
    'MA coefficients are not significant, so we remove them, (0 1 [1 4]).', ...
    newline,' ',newline, 'We now look at lage 18 and try to address it by ', ...
    'increasing the seasonal ARIMA. We have tried several specifications. ', ...
    'An extensive one would be (2 1 2), but the ACF and PACF do not improve.', ...
    newline,' ',newline]);

% remove TRAMO and significance testing in regression, fix outliers
s = makespec(spec3,'automdl',[],[], 'regression','aictest',[], 'NO OUTLIERS', ...
    'regression','variables','(LS1979.May AO1995.Jan easter[15] labor[8] thank[1])');

arima = travel3.arima;
s = x13spec(s,'arima','model',arima, 'series','name',arima);
t0 = x13(travel.dates,travel.data,s,'quiet');

arima = '(4 1 0)(0 1 1)';
s = x13spec(s,'arima','model',arima, 'series','name',arima);
t1 = x13(travel.dates,travel.data,s,'quiet');

arima = '(0 1 4)(0 1 1)';
s = x13spec(s,'arima','model',arima, 'series','name',arima);
t2 = x13(travel.dates,travel.data,s,'quiet');

arima = '(0 1 [1 4])(0 1 1)';
s = x13spec(s,'arima','model',arima, 'series','name',arima);
t3 = x13(travel.dates,travel.data,s,'quiet');

arima = '(0 1 [1 4])(2 1 2)';
s = x13spec(s,'arima','model',arima, 'series','name',arima);
t4 = x13(travel.dates,travel.data,s,'quiet');

fh = figure('Position',size10);
plot(fh,t0,t1,t2,t3,t4,'acf','pcf');

report(['Our final specification is therefore (0 1 [1 4])(0 1 1), which ', ...
    'gives us an almost perfectly clean ACF and PACF an acceptable spectrum ', ...
    'of the residuals.', newline]);

arima = '(0 1 [1 4])(0 1 1)';
spec4 = x13spec(s,'arima','model',arima,'series','name',travel.descr);
travel4 = x13(travel.dates,travel.data,spec4,'quiet');
```

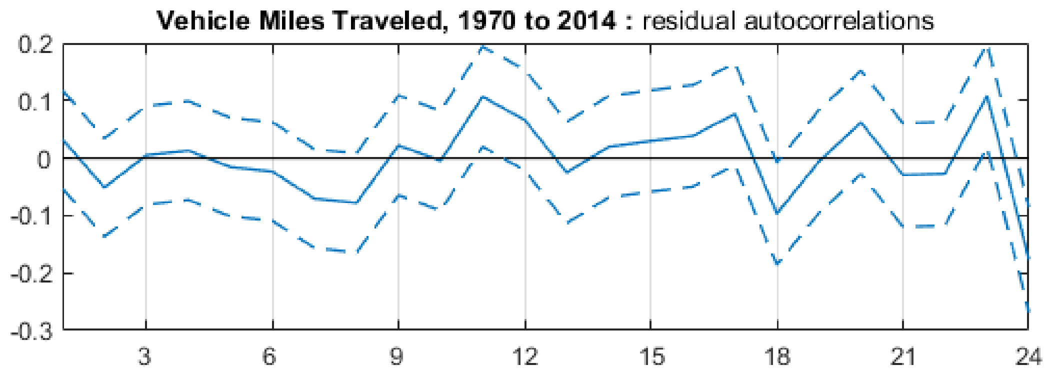
```
fh = figure('Position',sizeFig(50,80));
plot(fh,travel4,'acf','pcf','spr','rowwise')
```

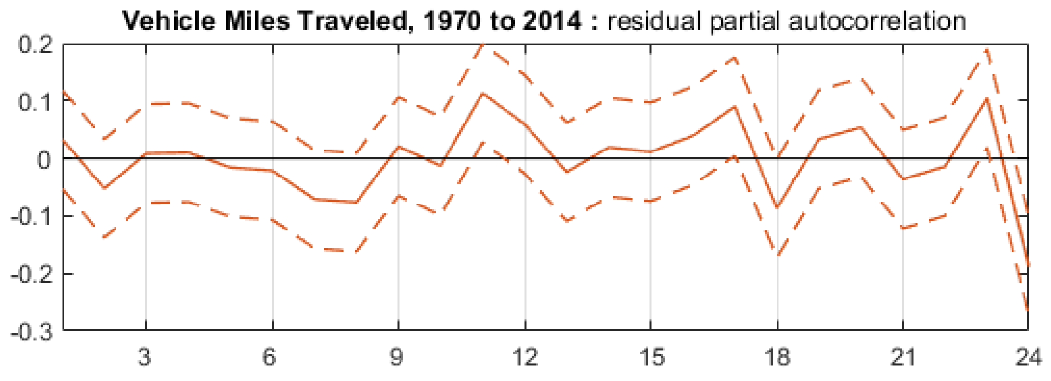
```
--------------------------------------------------------------------------------
 Step 4: Tweaking the ARIMA

 The Spectrum and autocorrelation problems have still not been resolved. It
 seems that we have to tweak the ARIMA specification manually. Inspecting the
 ACF, there is a significant problem at lag 4 and maybe at lag 18. Such a long
 lag (18) would normally not be an issue, but it might indicate an
 inappropriate choice of the seasonal part of the ARIMA, since 18 lags is 1.5
 years.

 But we address the problem at lag 4 first. We check (4 1 0) and (0 1 4) and
 find that (0 1 4) works much better.

 However, looking at the regression output, we notice that the 2nd and 3rd MA
 coefficients are not significant, so we remove them, (0 1 [1 4]).
```

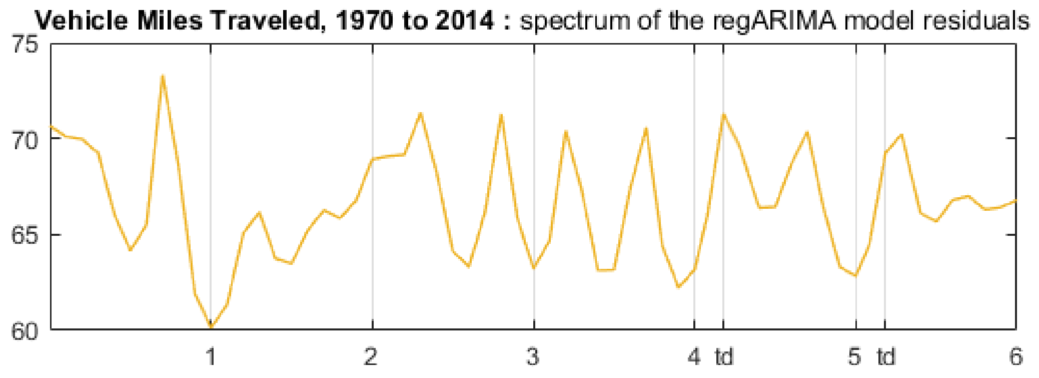


We now look at lag 18 and try to address it by increasing the seasonal ARIMA. We have tried several specifications. An extensive one would be (2 1 2), but the ACF and PACF do not improve.

Our final specification is therefore (0 1 [1 4])(0 1 1), which gives us an almost perfectly clean ACF and PACF and acceptable spectrum of the residuals.



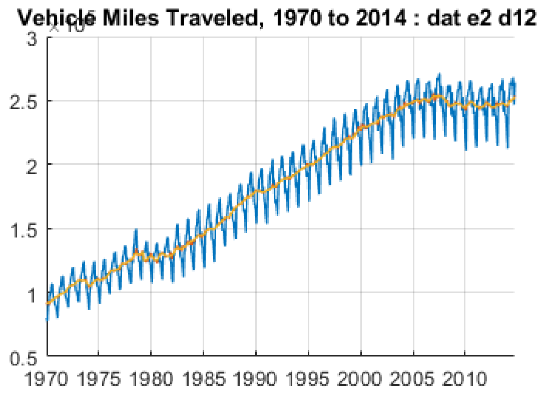


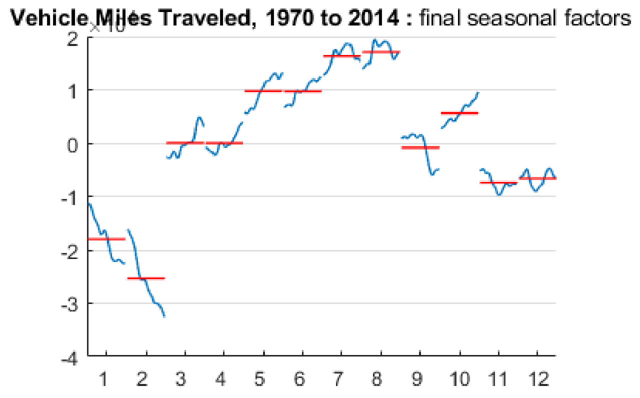


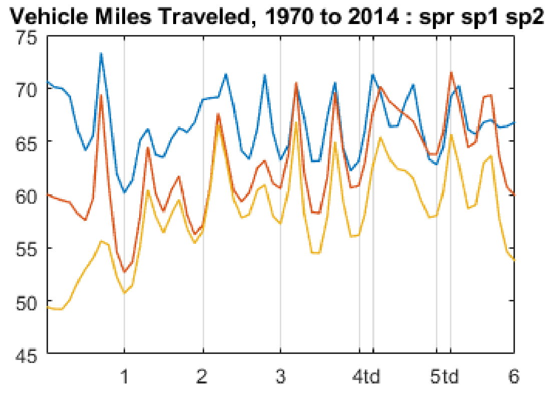
Step 5: Do the Seasonal Filtering

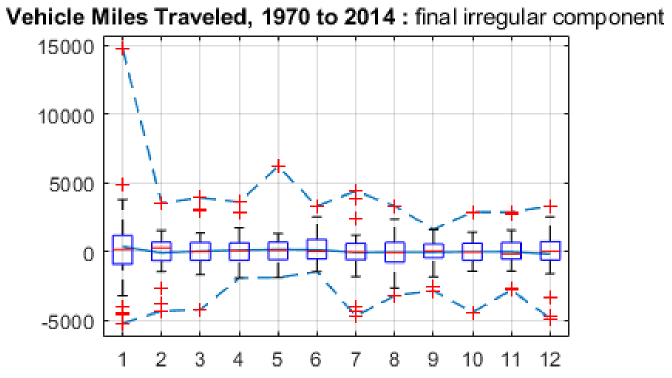
```
disp(sline)
fprintf(' Step 5: Performing the seasonal filtering\n\n');

spec5 = makespec(spec4,'X11','x11','mode','add');
travel5 = x13([travel.dates,travel.data],spec5);

figure('Position',size4)
ax = subplot(2,2,1);
plot(ax,travel5,'dat','e2','d12','comb');
ax = subplot(2,2,2);
plot(ax,travel5,'d10','bymonth');
ax = subplot(2,2,3);
plot(ax,travel5,'spr','sp1','sp2','comb');
ax = subplot(2,2,4);
plot(ax,travel5,'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,travel5);

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.']);
```

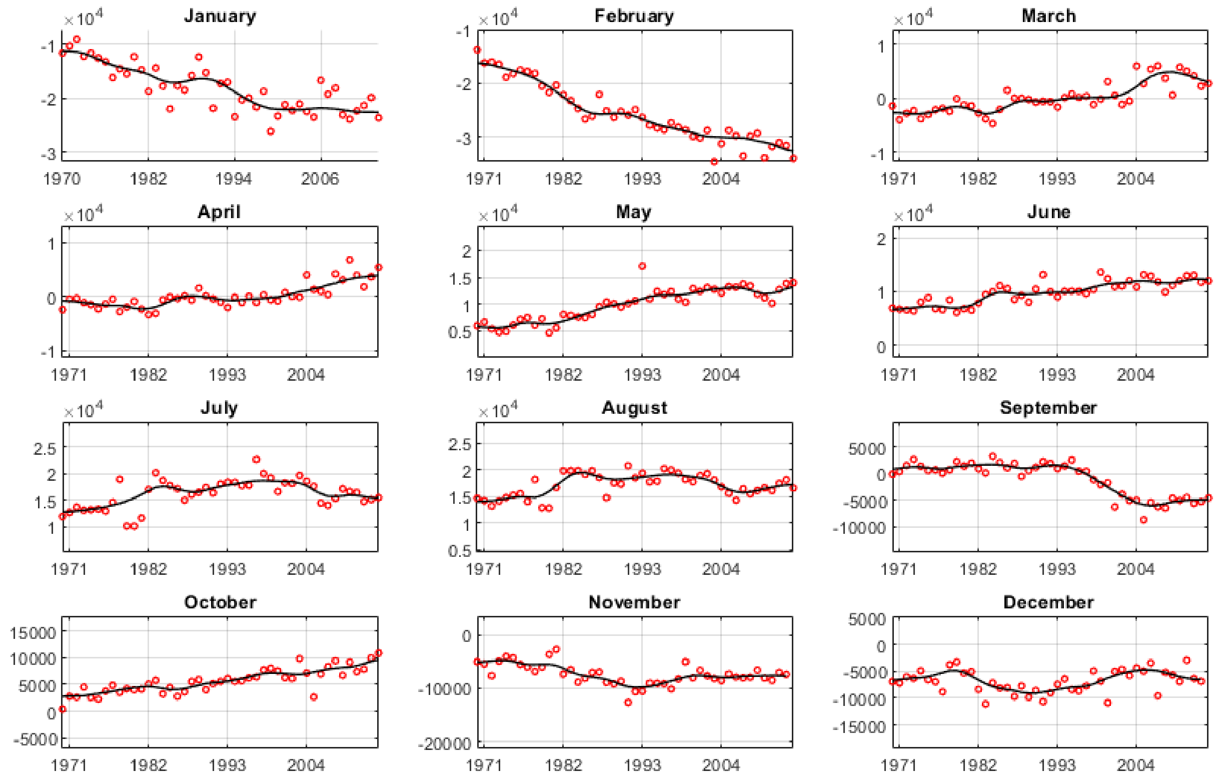
```
------------------------------------------------------------------------------
 Step 5: Performing the seasonal filtering

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


Vehicle Miles Traveled, 1970 to 2014 : dat e2 d12
×10^5
1970 1975 1980 1985 1990 1995 2000 2005 2010


Vehicle Miles Traveled, 1970 to 2014 : final seasonal factors
×10^4


Vehicle Miles Traveled, 1970 to 2014 : spr sp1 sp2
1 2 3 4td 5td 6


Vehicle Miles Traveled, 1970 to 2014 : final irregular component

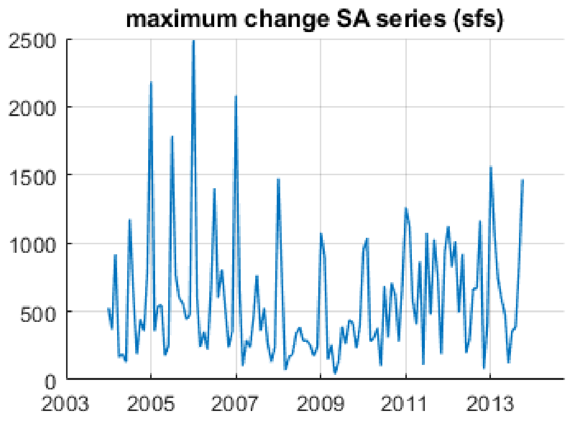

January
February
March
April
May
June
July
August
September
October
November
December

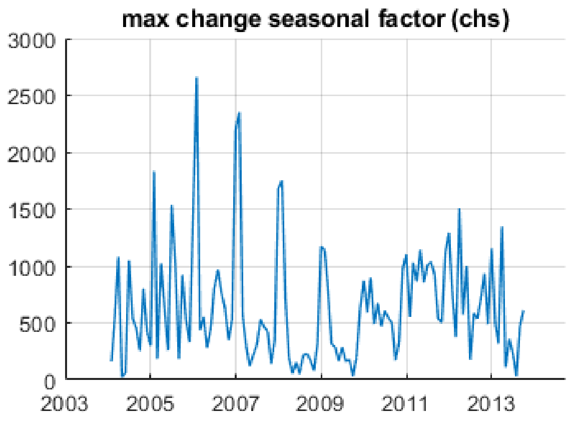
Step 6: Check Stability

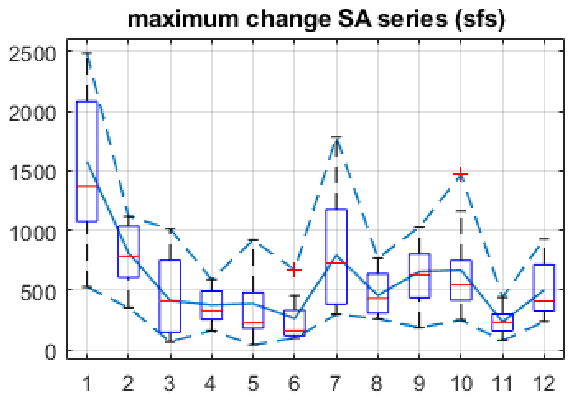
```
disp(sline)
fprintf(' Step 6: Checking stability (this takes a while...)\n\n');

spec6 = makespec(spec5,'SLIDING','HISTORY');
travel6 = x13([travel.dates,travel.data],spec6,'quiet');

% --- sliding span analysis

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

ax = subplot(2,2,1);
[~,ax] = plot(ax,travel6,'sfs','selection',[0 0 0 0 1]);
title(ax,'\bfmaximum change SA series (sfs)');
ax = subplot(2,2,2);
[~,ax] = plot(ax,travel6,'chs','selection',[0 0 0 0 1]);
title(ax,'\bfmax change seasonal factor (chs)');

ax = subplot(2,2,3);
[~,ax] = plot(ax,travel6,'sfs','selection',[0 0 0 0 1],'span','boxplot');
title(ax,'\bfmaximum change SA series (sfs)');
ax = subplot(2,2,4);
[~,ax] = plot(ax,travel6,'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,travel6,'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,travel6,'sar','span','boxplot')
title(ax,{'\bfmax % change of final vs','concurrent SA series (sar)'});

ax = subplot(2,2,2);
plot(ax,travel6,'sar','from',datenum(1985,1,1))
title(ax,{'\bf... since 1985'});
ax = subplot(2,2,4);
plot(ax,travel6,'sar','span','boxplot','from',datenum(1985,1,1))
title(ax,{'\bf... since 1985'});
```

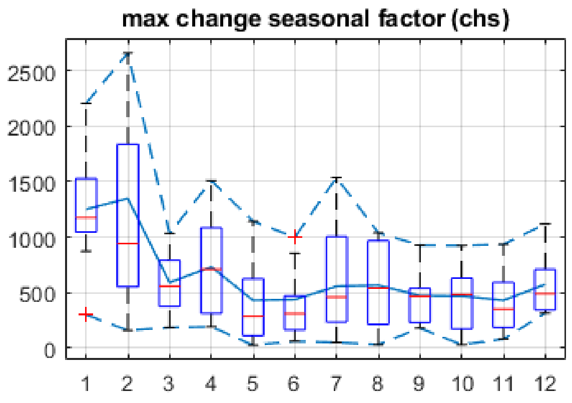
```
report(['CONCLUSION: The historical analysis also reveals small ', ...
    'changes of the seasonally adjusted series, except in the ', ...
    'beginning of the sample in the late 70s, early 80s.']);
```

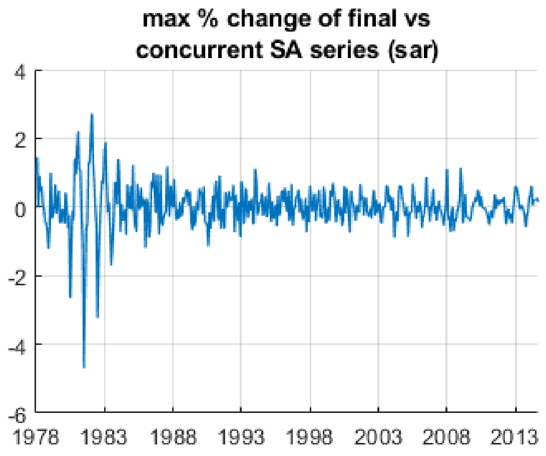
```
--------------------------------------------------------------------------------
 Step 6: Checking stability (this takes a while...)

 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.
```

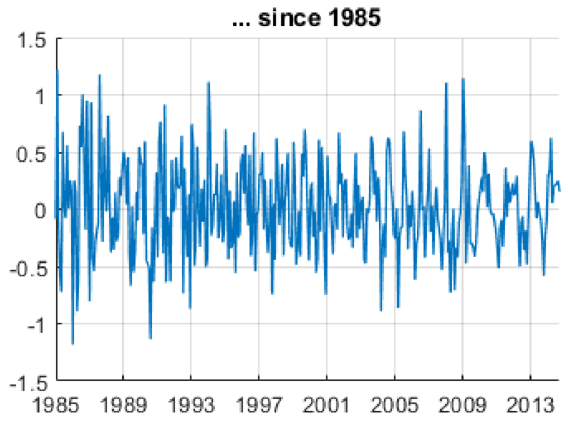


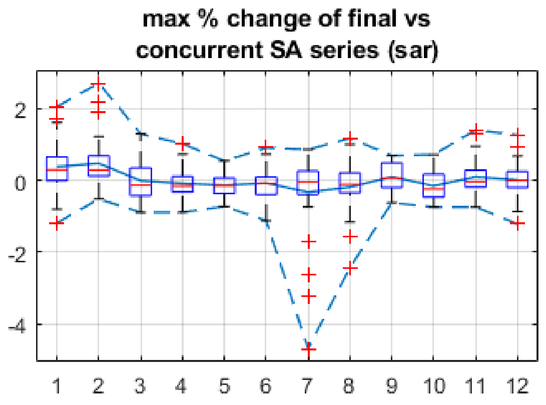


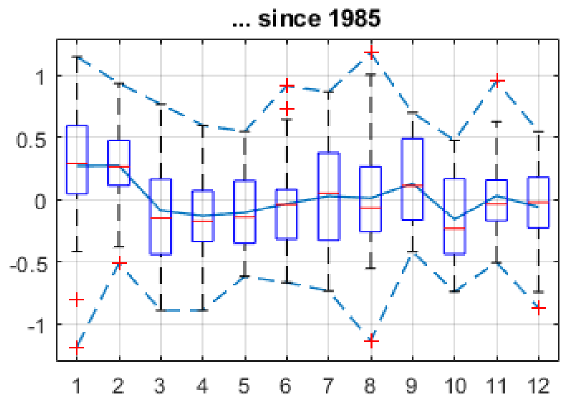





max % change of final vs
concurrent SA series (sar)
4
2
0
-2
-4
-6
1978
1983
1988
1993
1998
2003
2008
2013


... since 1985
1.5
1
0.5
0
-0.5
-1
-1.5
1985
1989
1993
1997
2001
2005
2009
2013


max % change of final vs
concurrent SA series (sar)
2
0
-2
-4
1
2
3
4
5
6
7
8
9
10
11
12

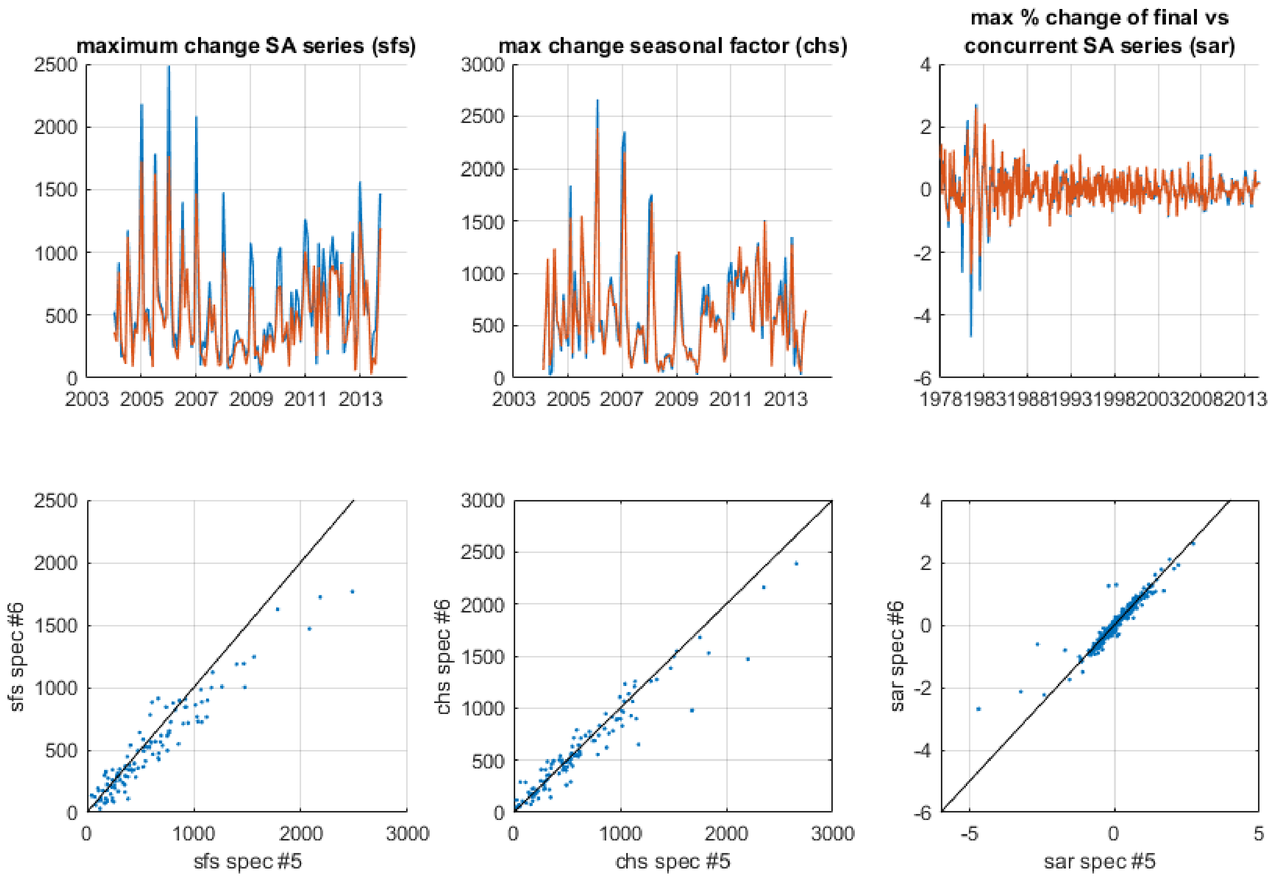

... since 1985
1
0.5
0
-0.5
-1
1
2
3
4
5
6
7
8
9
10
11
12

Step 7: Adjust Length Of Filter

```
disp(sline)
fprintf(' Step 7: Adjusting the length of the seasonal filter\n\n');

disp(travel6.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.']);

spec7 = makespec(spec6,'x11','seasonalma','s3x5');
travel7 = x13([travel.dates,travel.data],spec7,'quiet');

% --- sliding span and stability analysis

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

ax = subplot(2,3,1);
[~,ax] = plot(ax,travel6,travel7,'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(travel6.sfs);
fn6 = fieldnames(travel7.sfs);
ax = subplot(2,3,4);
scatter(ax,travel6.sfs.(fn5{end}),travel7.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,travel6,travel7,'chs','selection',[0 0 0 0 1],'comb');
title(ax,'\bfmax change seasonal factor (chs)');

fn5 = fieldnames(travel6.chs);
fn6 = fieldnames(travel7.chs);
ax = subplot(2,3,5);
plot(ax,travel6.chs.(fn5{end}),travel7.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,travel6,travel7,'sar','comb');
title(ax,{'\bfmax % change of final vs','concurrent SA series (sar)'});
```

```
ax = subplot(2,3,6);
plot(ax,travel6.sar.SA_revision,travel7.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 small, but the largest deviations for ', ...
    'are made a bit smaller with spec #7, so we keep this.']);
```

```
--------------------------------------------------------------------------------
 Step 7: Adjusting the length of the seasonal filter

 D 9.A  Moving seasonality ratio
 ----------------------------------------------------------------
               Jan       Feb       Mar       Apr       May       Jun
 ----------------------------------------------------------------
   I       1144.897  967.537  980.580 1095.617  893.605  937.510
   S        335.489  385.304  277.560  217.950  217.555  188.425
 RATIO        3.413    2.511    3.533    5.027    4.107    4.975

 ----------------------------------------------------------------
               Jul       Aug       Sep       Oct       Nov       Dec
 ----------------------------------------------------------------
   I        893.515 1001.450  956.141  968.932  921.431 1018.555
   S        267.326  294.939  255.364  198.975  203.795  273.814
 RATIO        3.342    3.395    3.744    4.870    4.521    3.720

 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 small, but the largest deviations for are made a bit
 smaller with spec #7, so we keep this.
```



maximum change SA series (sfs)
max change seasonal factor (chs)
max % change of final vs concurrent SA series (sar)
sfs spec #6
sfs spec #5
chs spec #6
chs spec #5
sar spec #6
sar spec #5

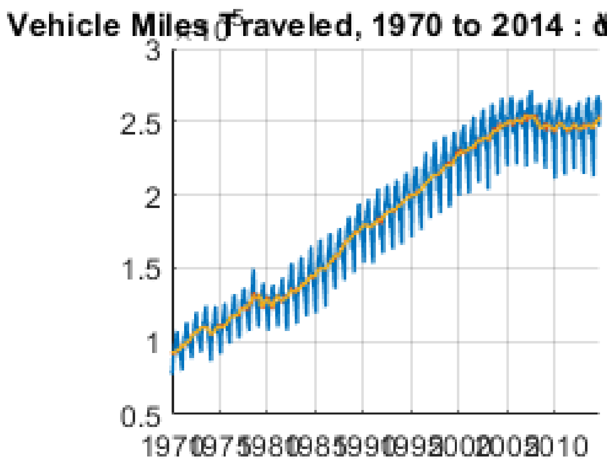
Final Step: specification for production

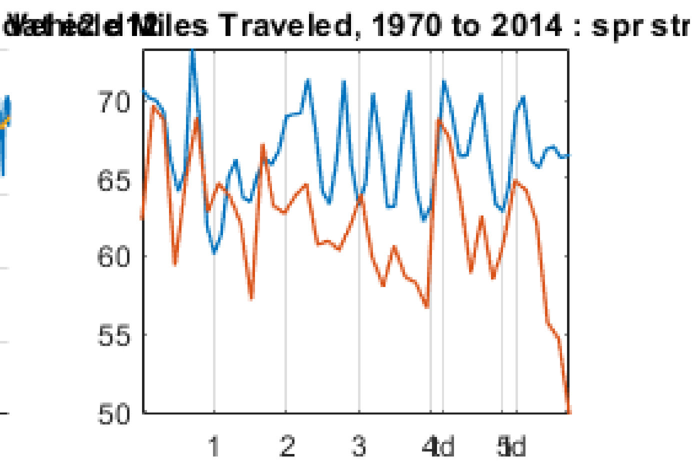
```
% remove history and sliding spans
spec8 = makespec(spec7, 'history',[],[], 'slidingspans',[],[]);

% this is the final specification
specfinal = makespec('series','name',travel.descr, 'DIAG', ...
    'NO OUTLIERS', 'regression','variables', ...
        '(AO1995.Jan LS1979.May easter[15] labor[8] td thank[1])', ...
    'regression','save','(hol td ao ls)', ...
    'arima', 'model', '(0 1 [1 4])(0 1 1)', ...
    'transform','function','none', ...
    'X11', 'x11','mode','add', 'x11','seasonalma','s3x5');

% compare the two
%disp(spec8);
disp(specfinal);

% perform the computations
travelsa = x13([travel.dates,travel.data],specfinal,'quiet');
travelhtml = x13([travel.dates,travel.data],specfinal,'html','quiet');
report('You can view results with web(travelhtml.out).')

% report final results

disp(travelsa);

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

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

fh = figure('Position',size6,'name','breaks');
seasbreaks(fh,travelsa);

fh = figure('Position',size4,'name','final decomposition');
plot(fh,travelsa,'d12','e2','d10','e3');

disp(travelsa.table('f3'));

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

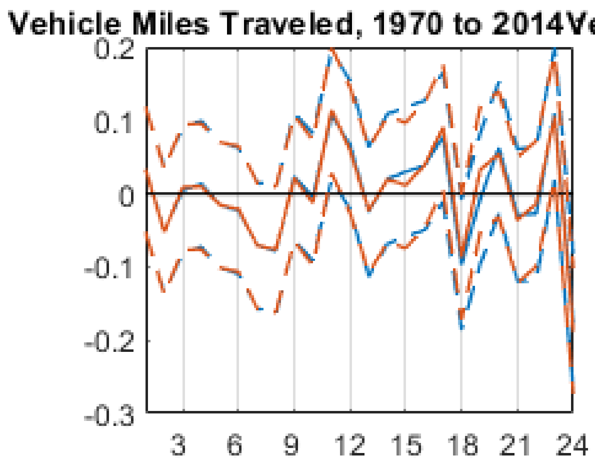
```
disp(dline);
```

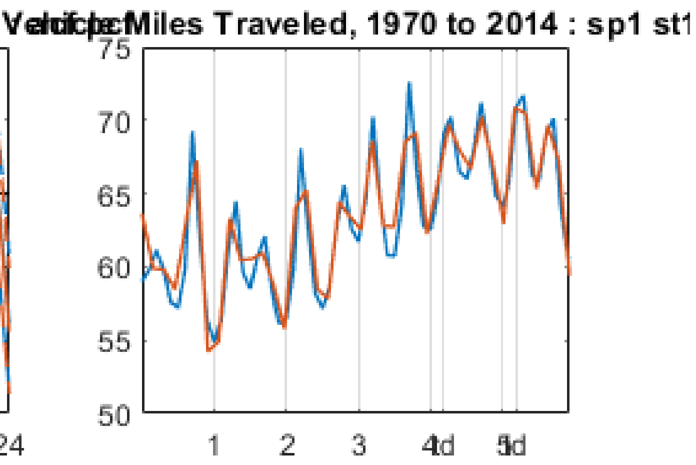
```
==============================================================================
 X-13/X-12 specification object
..............................................................................
 - series
    └─ name : Vehicle Miles Traveled, 1970 to 2014
 - arima
    └─ model : (0 1 [1 4])(0 1 1)
 - check
    ├─ save : (acf ac2 pcf)
    └─ print : (hst nrm)
 - estimate
    ├─ save : (mdl ref rsd rts est lks)
    └─ print : (est lks rts)
 - outlier
    └─ types : none
 - regression
    ├─ variables : (AO1995.Jan LS1979.May easter[15] labor[8] td thank[1])
    └─ save : (ao hol ls td)
 - spectrum
    ├─ save : (sp0 sp1 sp2 s1s s2s spr is0 is1 is2 st0 st1 st2 t1s t2s str it0
    |  it1 it2)
    └─ print : tpk
 - transform
    └─ function : none
 - x11
    ├─ print : (d8f d9a f2 f3 rsf)
    ├─ save : (b1 d10 d11 d12 d13 d16 d8 e2 e3)
    ├─ mode : add
    └─ seasonalma : s3x5

 You can view results with web(travelhtml.out).
==============================================================================
 X-13ARIMA-SEATS
 Version Number 1.1 Build 57 (x13as.exe)
..............................................................................
 Title : Vehicle Miles Traveled, 1970 to 2014
 Span  : 1970.1 to 2014.10, monthly data
 Data  : 538 observations
 Model : (0 1 [1 4])(0 1 1)
..............................................................................
 Time Series
 - ao  : regARIMA additive (or point) outlier factors (table A8.AO)
 - dat : unfiltered data
 - 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    8 ARMA iterations,   91 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:        2.06
        f2i : F 2.I:                                                   Sta
         f3 : F 3. Monitoring and Quality Assessment Statistics
```

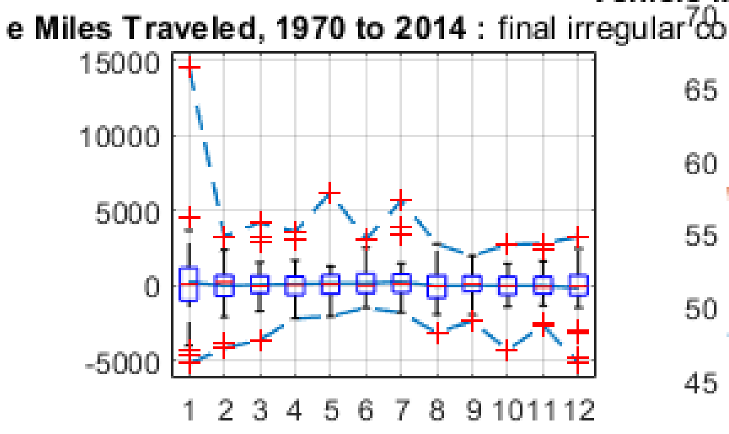
```
     tukey : Peak probabilities for Tukey spectrum estimator
.............................................................................
 NOTE: Use obj.table('name') to see content of a table, where 'name' can be
 abbreviated.
.............................................................................
 Time of run: 28-Apr-2021 14:02:19 (0.7 sec)
=============================================================================

 Series name: VehicleMilesTraveled_1970To201
 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):            837.131
 F-test for Moving Seasonality (D8):              7.080
 Identifiable Seasonality present

 Relative Contribution of the Irregular to the Variance

              I       C       S       P       TD    TOTAL
 F2.F    :    0.34   37.17   63.51    4.18    0.25  105.44

 I/C RATIO:         2.06  I/S RATIO:         3.78

 M1:   0.040  M2:   0.036  M3:   0.531  M4:   0.132
 M5:   0.484  M6:   0.089  M7:   0.130  M8:   0.195
 M9:   0.107  M10:  0.158  M11:  0.144
 Q =  0.19
 Q(without M2) =  0.21

 Visually distinct seasonal spectral peaks were not found.

 Visually distinct trading day spectral peaks were found in:
     regARIMA model residuals

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

                                         S1       S2       S3       S4       S5       S6       TD
                                       ------   ------   ------   ------   ------   ------   ------
  Model Residuals                       0.717    0.426    0.880    0.005    0.303    0.241    0.634
  Prior Adjusted Series (Table B1)      0.960*   0.999**  0.998**  0.988*   1.000**  0.818    0.024
  Seasonally adjusted series (E2)       0.045    0.034    0.112    0.053    0.033    0.125    0.864
  Modified Irregular (E3)               0.303    0.070    0.115    0.050    0.033    0.127    0.884
  ----------
     ** - Peak Probability > 0.99,
      * - 0.90 < Peak Probability < 0.99
  F 3. Monitoring and Quality Assessment Statistics
       All the measures below are in the range from 0 to 3 with an
       acceptance region from 0 to 1.

    1. The relative contribution of the irregular over three      M1  =  0.040
       months span (from Table F 2.B).

    2. The relative contribution of the irregular component       M2  =  0.036
       to the stationary portion of the variance (from Table
       F 2.F).

    3. The amount of month to month change in the irregular       M3  =  0.531
       component as compared to the amount of month to month
       change in the trend-cycle (from Table F2.H).

    4. The amount of autocorrelation in the irregular as          M4  =  0.132
       described by the average duration of run (Table F 2.D).

    5. The number of months it takes the change in the trend-     M5  =  0.484
       cycle to surpass the amount of change in the irregular
       (from Table F 2.E).

    6. The amount of year to year change in the irregular as      M6  =  0.089
       compared to the amount of year to year change in the
       seasonal (from Table F 2.H).

    7. The amount of moving seasonality present relative to       M7  =  0.130
       the amount of stable seasonality (from Table F 2.I).

    8. The size of the fluctuations in the seasonal component     M8  =  0.195
       throughout the whole series.

    9. The average linear movement in the seasonal component      M9  =  0.107
       throughout the whole series.

   10. Same as 8, calculated for recent years only.               M10 =  0.158

   11. Same as 9, calculated for recent years only.               M11 =  0.144

          *** ACCEPTED *** at the level   0.19

          *** Q (without M2) =  0.21  ACCEPTED.

 CONCLUSION: The decomposition appears acceptable.
===============================================================================
```

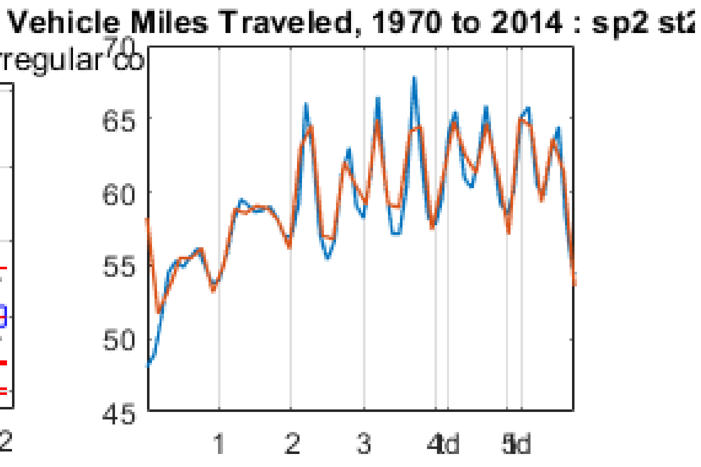

Vehicle Miles Traveled, 1970 to 2014 : d
×10^5
3
2.5
2
1.5
1
0.5
1970 1975 1980 1985 1990 1995 2000 2005 2010

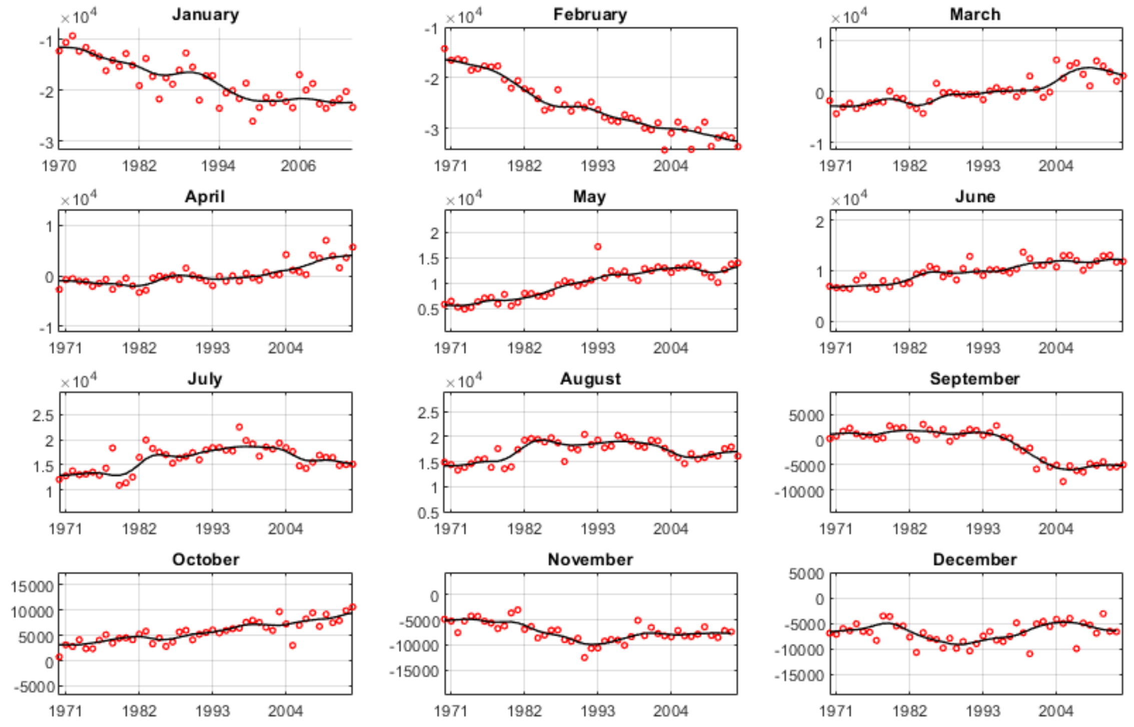

Vehicle Miles Traveled, 1970 to 2014 : spr str
70
65
60
55
50
1
2
3
4
4d
5
5d

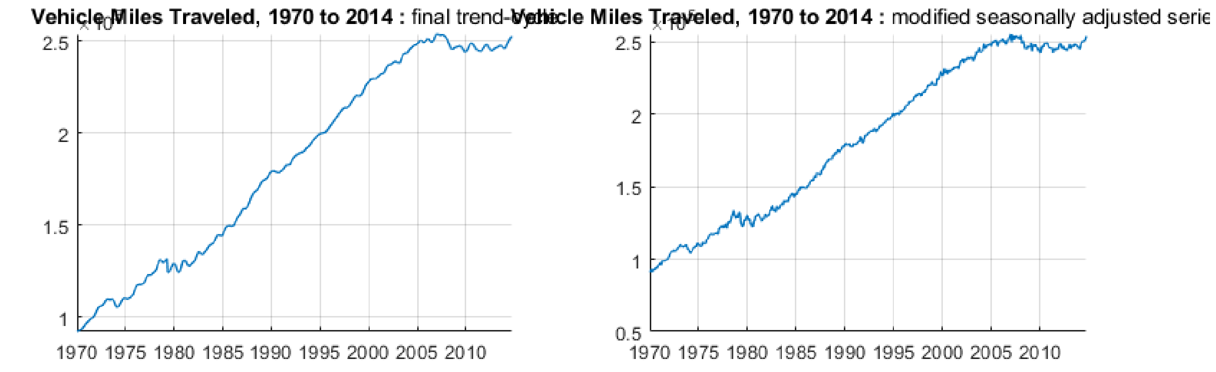

Vehicle Miles Traveled, 1970 to 2014
0.2
0.1
0
-0.1
-0.2
-0.3
3 6 9 12 15 18 21 24

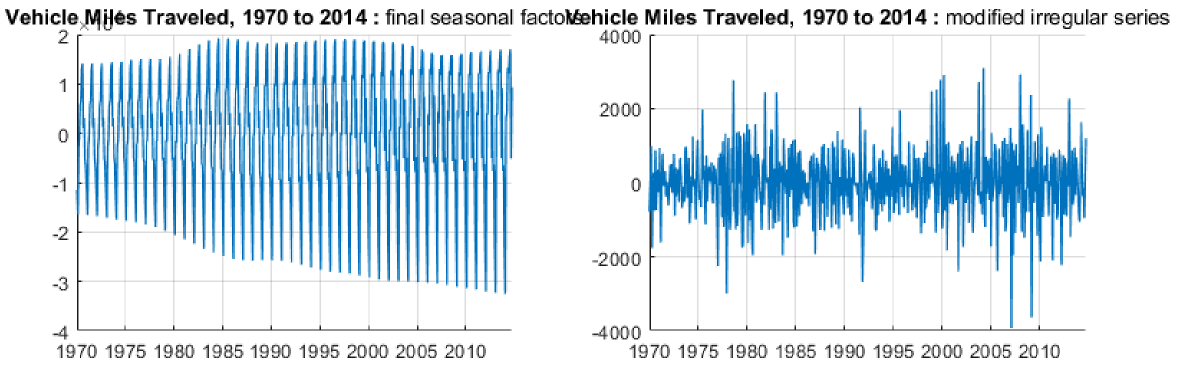

Vehicle Miles Traveled, 1970 to 2014 : sp1 st1
75
70
65
60
55
50
1
2
3
4
4d
5
5d


e Miles Traveled, 1970 to 2014 : final irregular co
15000
10000
5000
0
-5000
1 2 3 4 5 6 7 8 9 10 11 12


Vehicle Miles Traveled, 1970 to 2014 : sp2 st2
70
65
60
55
50
45
1
2
3
4
4d
5
5d







Published with MATLAB® R2020a

COMPARING VARIOUS ALGORITHMS

Load Data and Specs

```
fprintf(['We are performing the seasonal decomposition with eight different ', ...
    'algorithms and compare the outcome.\n\n']);

% These are monthly data taken from the book by  D. Ladiray and
% B. Quenneville on the X-11 method.
load LadirayQuenneville

% This is a classic dataset used in ARIMA modelling and seasonal
% adjustment. It is known as Box-Jenkin's Series G and decribes the number
% of airline passengers from Jan 1949 and Dec 1960.
load BoxJenkinsG

% Unemployment quota in the USA.
load unemp;

% Unemployment quota in the USA.
load travel;

% choose the data set to use
data = BoxJenkinsG;
%data = LadirayQuenneville;
%data = unemp;
%data = travel;
```

```
basespec = makespec('LOG','TDAYS','EASTER','SPECTRUM');
```

```
We are performing the seasonal decomposition with eight different algorithms and compare the outcome.
```

Computations

```
n = {};

n{end+1} = 'X-13AS with X-11'; disp(n{end})
spec = makespec(basespec,'TRAMO','X11','series','title',n{end});
%spec = makespec(basespec,'series','title',n);
x3x = x13(data.dates,data.data,spec,'quiet');
x3x.addMatlabSpectrum;

n{end+1} = 'X-13AS with SEATS'; disp(n{end})
spec = makespec(basespec,'TRAMO','SEATS','series','title',n{end});
%spec = makespec(basespec,'SEATS','series','title',n);
x3s = x13(data.dates,data.data,spec,'quiet');
x3s.addMatlabSpectrum;

n{end+1} = 'X-12'; disp(n{end})
spec = makespec(basespec,'TRAMO','X11','series','title',n{end});
%spec = makespec(basespec,'series','title',n);
x2 = x13(data.dates,data.data,spec,'x-12','quiet');

n{end+1} = 'X-11'; disp(n{end})
spec = makespec(basespec,'X11','series','title',n{end});
%spec = makespec(basespec,'series','title',n);
x1 = x13(data.dates,data.data,spec,'x-11','quiet');

n{end+1} = 'Method I'; disp(n{end})
spec = makespec(basespec,'X11','series','title',n{end});
%spec = makespec(basespec,'series','title',n);
m = x13(data.dates,data.data,spec,'method1','quiet');

n{end+1} = 'FIXED'; disp(n{end})
spec = makespec(basespec,'FIXEDSEASONAL','series','title',n{end});
f = x13(data.dates,data.data,spec,'quiet');

n{end+1} = 'C.A.M.P.LE.T'; disp(n{end})
spec = makespec(basespec,'CAMPLET','series','title',n{end});
c = x13(data.dates,data.data,spec,'quiet');

n{end+1} = 'seas.m'; disp(n{end})
spec = x13spec(basespec,'series','title',n{end});
s = x13(data.dates,data.data,spec,'prog','seas.m','quiet');
```

```
X-13AS with X-11
X-13AS with SEATS
X-12
X-11
Method I
FIXED
C.A.M.P.LE.T
seas.m
```

Correlation between seasonal factors

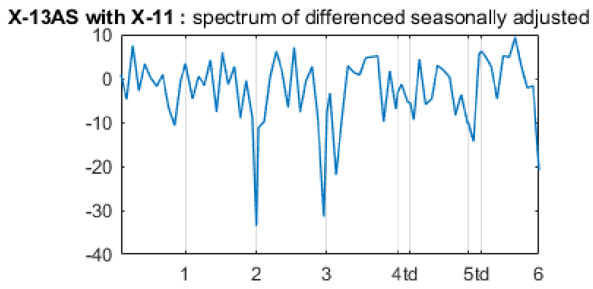
```
fprintf(' Correlations between different seasonally adjusted series\n\n');
all_sa = [x3x.d11.d11,x3s.s11.s11,x2.d11.d11,x1.d11.d11, ...
    m.d11.d11,f.sa.sa,c.sa.sa,s.sa.sa];
corr_sa = corr(all_sa);
corr_sa_tbl = table(corr_sa(:,1),corr_sa(:,2),corr_sa(:,3),corr_sa(:,4), ...
    corr_sa(:,5),corr_sa(:,6),corr_sa(:,7),corr_sa(:,8), ...
    'VariableNames',n, 'RowNames',n);
disp(corr_sa_tbl);

n(7) = [];  % CAMPLET does not produce a seasonal factor
fprintf(' Correlations between different seasonal factors\n\n');
all_sf = [x3x.d10.d10,x3s.s10.s10,x2.d10.d10,x1.d10.d10, ...
    m.d10.d10,f.sf.sf,s.sf.sf];
corr_sf = corr(all_sf);
corr_sf_tbl = table(corr_sf(:,1),corr_sf(:,2),corr_sf(:,3),corr_sf(:,4), ...
    corr_sf(:,5),corr_sf(:,6),corr_sf(:,7), ...
    'VariableNames',n, 'RowNames',n);
disp(corr_sf_tbl);

fprintf('The differences are very small.\n\n');
```

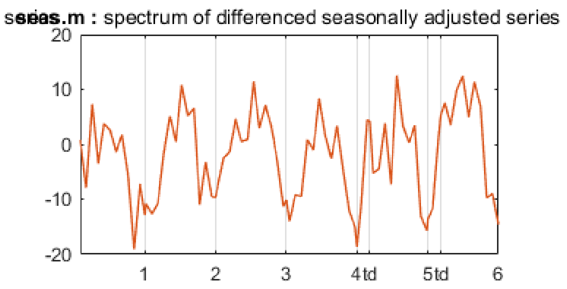
```
 Correlations between different seasonally adjusted series

                         X-13AS with X-11    X-13AS with SEATS     X-12       X-11     Method I     FIXED     C.A.M.P.LE.T     seas.m 
                         ________________    _________________    _______    _______    ________    _______    ____________    _______

    X-13AS with X-11                1               0.9999        0.99997    0.99983     0.9998     0.99814        0.9993       0.9997
    X-13AS with SEATS          0.9999                    1        0.99987    0.99985     0.9999     0.99814       0.99943       0.9998
    X-12                      0.99997              0.99987              1     0.9998    0.99978     0.99814       0.99925      0.99967
    X-11                      0.99983              0.99985         0.9998          1    0.99992     0.99784       0.99932      0.99983
    Method I                   0.9998               0.9999        0.99978    0.99992          1     0.99779       0.99947      0.99993
    FIXED                     0.99814              0.99814        0.99814    0.99784    0.99779           1       0.99737      0.99743
    C.A.M.P.LE.T               0.9993              0.99943        0.99925    0.99932    0.99947     0.99737             1      0.99943
    seas.m                     0.9997               0.9998        0.99967    0.99983    0.99993     0.99743       0.99943            1

 Correlations between different seasonal factors

                         X-13AS with X-11    X-13AS with SEATS     X-12       X-11     Method I     FIXED      seas.m 
                         ________________    _________________    _______    _______    ________    _______    _______

    X-13AS with X-11                1               0.9988        0.99985    0.99898    0.99821     0.98194    0.99661
    X-13AS with SEATS          0.9988                    1        0.99885    0.99868    0.99919     0.98262    0.99765
    X-12                      0.99985              0.99885              1    0.99903    0.99824      0.9829    0.99654
    X-11                      0.99898              0.99868        0.99903          1    0.99917     0.98046    0.99787
    Method I                  0.99821              0.99919        0.99824    0.99917          1        0.98    0.99912
    FIXED                     0.98194              0.98262         0.9829    0.98046       0.98           1    0.97637
    seas.m                    0.99661              0.99765        0.99654    0.99787    0.99912     0.97637          1

The differences are very small.
```

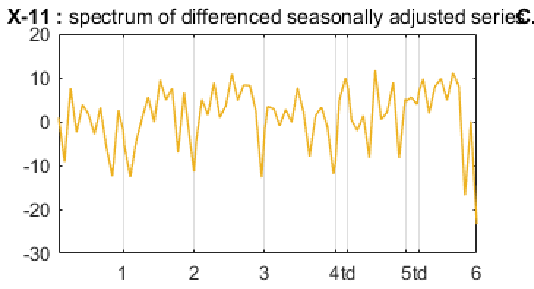
Charts

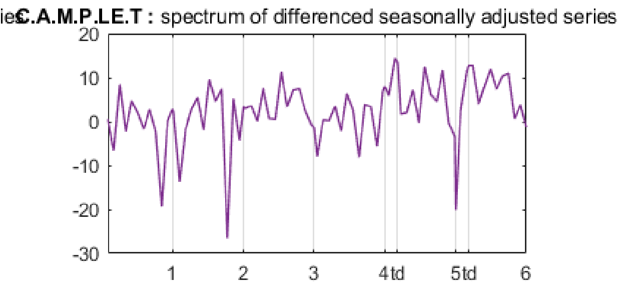
```
fh = figure('Position',[74 69 854 693]); movegui(fh,'center');
plot(fh,x3x,s,x1,c,m,f,'sp1')
%plot(x3x,x3s,'sp1','s1s','combined','quiet')

fh = figure('Position',[74 69 854 693]); movegui(fh,'center');
plot(fh,x3x,s,x1,c,m,f,'sp2','quiet')
%plot(x3x,x3s,'sp2','s2s','combined','quiet')

fh = figure('Position',[211 45 998 706]); movegui(fh,'center');
ah = subplot(3,2,1); plot(ah,x3x,x3s,'e2','s11','comb','quiet')
ah = subplot(3,2,3); plot(ah,x3x,x1,'e2','comb');
ah = subplot(3,2,5); plot(ah,x3x,m,'d11','comb');
ah = subplot(3,2,2); plot(ah,x3x,s,'d11','sa','comb','quiet')
ah = subplot(3,2,4); plot(ah,x3x,c,'d11','sa','comb','quiet')
ah = subplot(3,2,6); plot(ah,x3x,f,'d11','sa','comb','quiet')

fh = figure('Position',[211 45 998 706]); movegui(fh,'center');
ah = subplot(3,2,1); plot(ah,x3x,x3s,'d12','s12','comb','quiet')
ah = subplot(3,2,3); plot(ah,x3x,x1,'d12','comb');
ah = subplot(3,2,5); plot(ah,x3x,m,'d12','comb');
ah = subplot(3,2,2); plot(ah,x3x,s,'d12','tr','comb','quiet')
ah = subplot(3,2,4); plot(ah,x3x,c,'d12',c.keyv.tr,'comb','quiet')
ah = subplot(3,2,6); plot(ah,x3x,f,'d12','tr','comb','quiet')

fprintf(['CAMPLET is different because it is an only backward looking filter. ', ...
    'The advantage of a purely backward-looking algorithm is that the seasonal ', ...
    'adjustment is not changed when new data come along. But as can be seen, ', ...
    'the trend and seasonally adjusted series are merkedly different. To be ', ...
    'fair, they are estimated ``on the go,'' so without the benefit of the ', ...
    'future evolution, so we should expect larger differences.\n']);
```

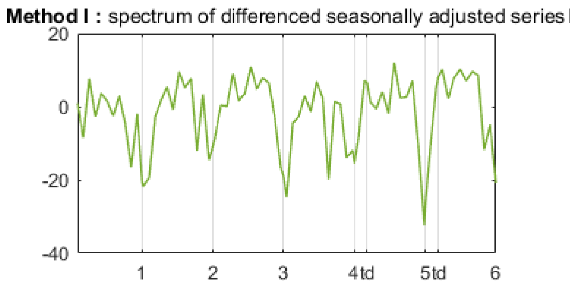
```
CAMPLET is different because it is an only backward looking filter. The advantage of a purely backward-
looking algorithm is that the seasonal adjustment is not changed when new data come along. But as can be
seen, the trend and seasonally adjusted series are merkedly different. To be fair, they are estimated
``on the go,' so without the benefit of the future evolution, so we should expect larger differences.
```

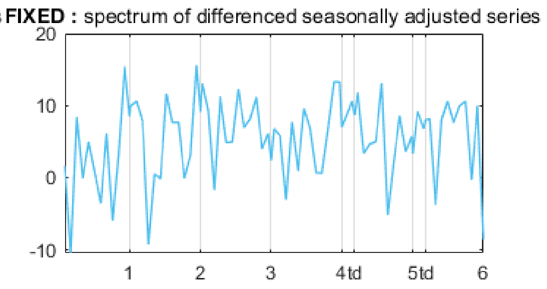

X-13AS with X-11 : spectrum of differenced seasonally adjusted series
10
0
-10
-20
-30
-40
1
2
3
4 td
5 td
6

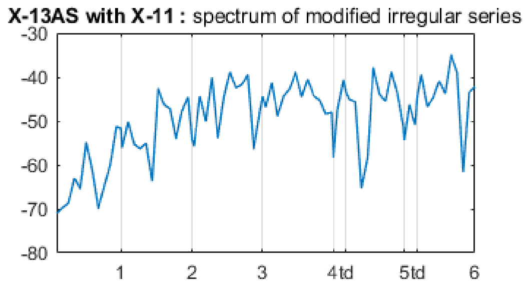

seas.m : spectrum of differenced seasonally adjusted series
20
10
0
-10
-20
1
2
3
4 td
5 td
6

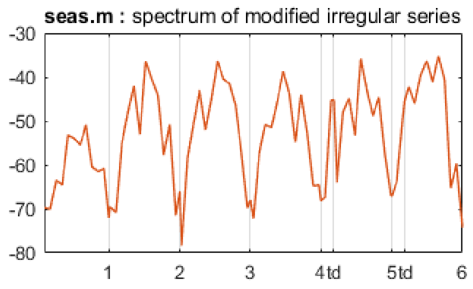

X-11 : spectrum of differenced seasonally adjusted series
20
10
0
-10
-20
-30
1
2
3
4 td
5 td
6

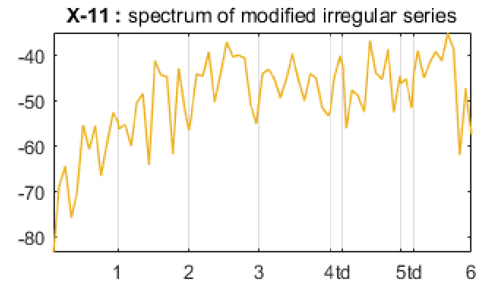

C.A.M.P.L.E.T : spectrum of differenced seasonally adjusted series
20
10
0
-10
-20
-30
1
2
3
4 td
5 td
6

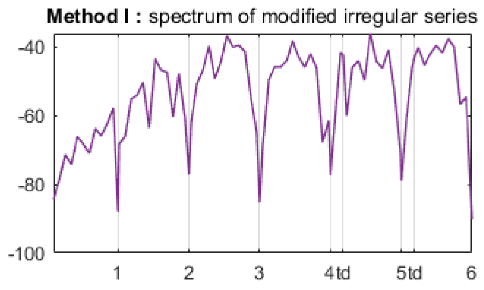

Method I : spectrum of differenced seasonally adjusted series
20
0
-20
-40
1
2
3
4 td
5 td
6

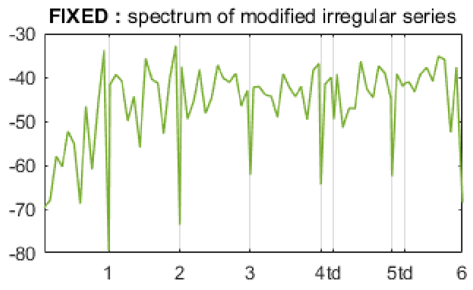

FIXED : spectrum of differenced seasonally adjusted series
20
10
0
-10
1
2
3
4 td
5 td
6


X-13AS with X-11 : spectrum of modified irregular series
-30
-40
-50
-60
-70
-80
1
2
3
4td
5td
6

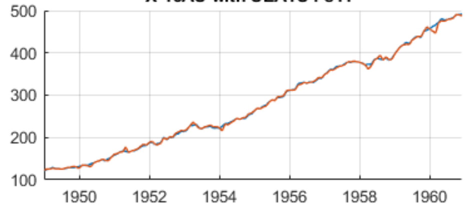

seas.m : spectrum of modified irregular series
-30
-40
-50
-60
-70
-80
1
2
3
4td
5td
6


X-11 : spectrum of modified irregular series
-40
-50
-60
-70
-80
1
2
3
4td
5td
6

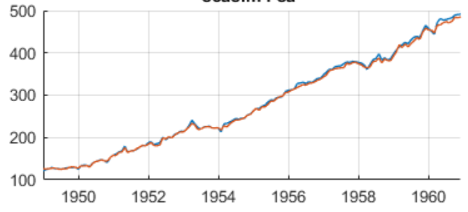

Method I : spectrum of modified irregular series
-40
-60
-80
-100
1
2
3
4td
5td
6


FIXED : spectrum of modified irregular series
-30
-40
-50
-60
-70
-80
1
2
3
4td
5td
6

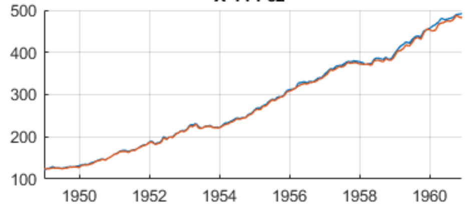
X-13AS with X-11 : e2
X-13AS with SEATS : s11



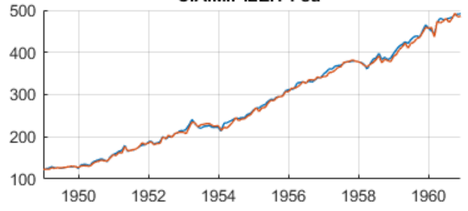
X-13AS with X-11 : d11
seas.m : sa



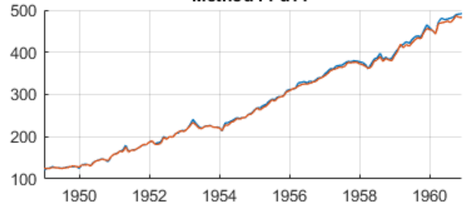
X-13AS with X-11 : e2
X-11 : e2



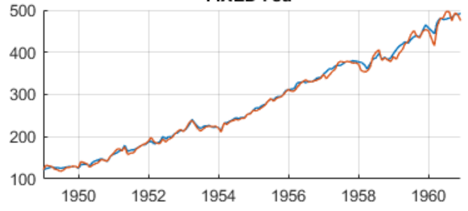
X-13AS with X-11 : d11
C.A.M.P.LE.T : sa



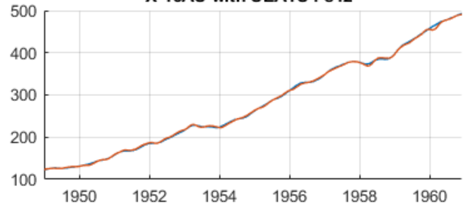
X-13AS with X-11 : d11
Method I : d11



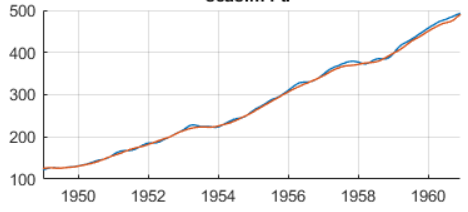
X-13AS with X-11 : d11
FIXED : sa



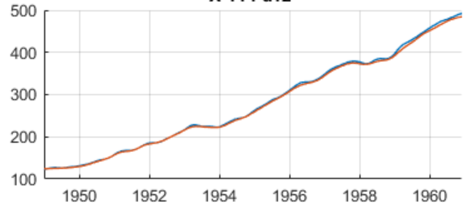
X-13AS with X-11 : d12
X-13AS with SEATS : s12



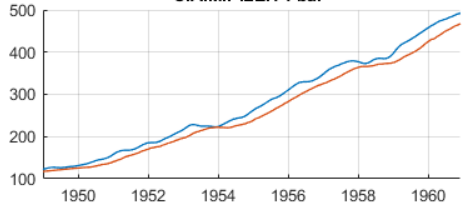
X-13AS with X-11 : d12
seas.m : tr



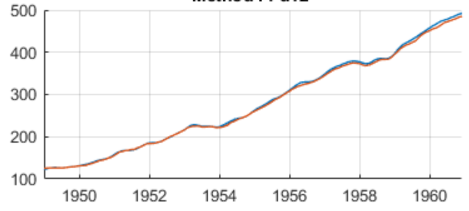
X-13AS with X-11 : d12
X-11 : d12



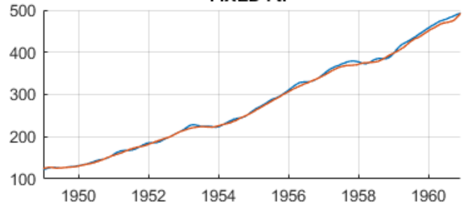
X-13AS with X-11 : d12
C.A.M.P.LE.T : bar



X-13AS with X-11 : d12
Method I : d12



X-13AS with X-11 : d12
FIXED : tr



Demo of a Composite Run

DEMO for X13 Toolbox: Composite Run

```
%#ok<*CHARTEN>
%#ok<*SAGROW>

% 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);
common = makespec('MULT','TRAMOPURE','X11','AO','LS','ACF', ...
        'series','comptype','add');
for c = 1:n
    spec{c} = x13spec(common, 'series','title',ctry{c});
end

% ... and for the composite series
compspec = makespec('MULT','TRAMOPURE','X11','AO','LS','ACF', ...
    'outlier','method','addone', ...
    'composite','save','(cms itn isa iir iaf)', ...
    'composite','title','Aggregate');

% run x13 and show result
xgdp = x13(dates,data,spec,compspec,'graphicsloc',grloc,'quiet');
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).
==============================================================================
 composite seasonal adjustemnt object
 Version Number 1.1 Build 57
 Data : 2000.3 to 2014.3
..............................................................................
 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: 28-Apr-2021 17:30:27 (17.1 sec)
==============================================================================
```

REPORT SOME RESULTS

```
allseries = xgdp.listofseries;

fprintf('\n --- KEY STATISTICS -----------------------------------------------------------\n');
% THERE IS A PROBLEM HERE. x2d is not available.
for c = 1:numel(allseries)
    try
        strOUT   = xgdp.(allseries{c}).x2d;
    catch
        strOUT = '';
    end
    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
    try
        for l = first:1:0
            disp(linesOUT{end+l});
        end
    end
end
fprintf('\n ------------------------------------------------------------------------------\n\n');

report(['Most M-statistics pass, with some exceptions. The M4 and M8 ', ...
    'statistic marginally fail (>1) for some countries countries, but ', ...
    ' these are not the most crucial quality control statistics, so we ', ...
    'ignore that. The M1-statistic  marginally fails for the United ', ...
    'Kingdom. We ignore this as well, but you are welcome to look for ', ...
    'improvements. Interestingly, TRAMO identifies an ARIMA model with ', ...
    'no seasonal component for UK (model (0 1 1)), yet stable seasonality ', ...
    'is present according to the tests reported in table d8a.']);
% 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: (0 1 1)(0 1 1)

identifiable seasonality present

M1:   0.080  M2:   0.029  M3:   0.000  M4:   0.413
M5:   0.200  M6:   0.662  M7:   0.086  M8:   0.250
M9:   0.148  M10:  0.336  M11:  0.247
Q =  0.20
Q(without M2) =  0.22

Comparison between Smoothness Measures for Direct and Indirect Adjustments

                   (Dir - Ind) Percentage Change
                  Entire srs.         Last 3 yrs.
  R1 - MSE              4.304               0.860
  R1 - RMSE             2.176               0.431
  R2 - MSE             92.592              22.479
  R2 - RMSE            72.782              11.954

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

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

identifiable seasonality present

M1:   0.185  M2:   0.088  M3:   0.038  M4:   0.785
M5:   0.200  M6:   0.128  M7:   0.176  M8:   0.324
M9:   0.260  M10:  0.317  M11:  0.317
Q =  0.23
Q(without M2) =  0.24

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

identifiable seasonality present

M1:   0.008  M2:   0.013  M3:   0.000  M4:   0.413
M5:   0.200  M6:   0.155  M7:   0.057  M8:   0.123
M9:   0.082  M10:  0.158  M11:  0.154
Q =  0.11
Q(without M2) =  0.12

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

identifiable seasonality present

M1:   0.029  M2:   0.017  M3:   0.000  M4:   0.661
M5:   0.200  M6:   0.471  M7:   0.072  M8:   0.232
M9:   0.167  M10:  0.276  M11:  0.235
Q =  0.16
```

```
Q(without M2) =  0.18

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

identifiable seasonality present

M1:   0.021  M2:   0.006  M3:   0.000  M4:   0.909
M5:   0.200  M6:   0.771  M7:   0.095  M8:   0.406
M9:   0.103  M10:  0.627  M11:  0.592
Q =  0.22
Q(without M2) =  0.25

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

identifiable seasonality present

M1:   0.029  M2:   0.003  M3:   0.000  M4:   0.661
M5:   0.200  M6:   0.744  M7:   0.142  M8:   0.386
M9:   0.164  M10:  0.277  M11:  0.255
Q =  0.18
Q(without M2) =  0.21

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

identifiable seasonality present

M1:   0.055  M2:   0.012  M3:   0.000  M4:   0.165
M5:   0.200  M6:   0.198  M7:   0.132  M8:   0.308
M9:   0.129  M10:  0.201  M11:  0.099
Q =  0.13
Q(without M2) =  0.14

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

identifiable seasonality present

M1:   0.128  M2:   0.096  M3:   0.070  M4:   0.413
M5:   0.200  M6:   0.399  M7:   0.169  M8:   0.448
M9:   0.334  M10:  0.518  M11:  0.487
Q =  0.25
Q(without M2) =  0.27

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

identifiable seasonality present

M1:   0.145  M2:   0.018  M3:   0.000  M4:   1.280
M5:   0.200  M6:   0.539  M7:   0.231  M8:   0.879
M9:   0.276  M10:  0.662  M11:  0.533
Q =  0.35
Q(without M2) =  0.39
```

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

identifiable seasonality present

M1:   0.163  M2:   0.096  M3:   0.297  M4:   1.156
M5:   0.200  M6:   0.481  M7:   0.112  M8:   0.222
M9:   0.142  M10:  0.244  M11:  0.240
Q =  0.28
Q(without M2) =  0.31

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

identifiable seasonality present

M1:   0.027  M2:   0.015  M3:   0.000  M4:   0.413
M5:   0.200  M6:   0.118  M7:   0.081  M8:   0.182
M9:   0.148  M10:  0.152  M11:  0.148
Q =  0.12
Q(without M2) =  0.13

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

identifiable seasonality present

M1:   0.303  M2:   0.177  M3:   0.113  M4:   0.413
M5:   0.200  M6:   0.077  M7:   0.159  M8:   0.326
M9:   0.297  M10:  0.227  M11:  0.194
Q =  0.21
Q(without M2) =  0.22

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

identifiable seasonality present

M1:   0.032  M2:   0.004  M3:   0.000  M4:   0.289
M5:   0.200  M6:   0.807  M7:   0.141  M8:   0.380
M9:   0.355  M10:  0.632  M11:  0.626
Q =  0.20
Q(without M2) =  0.22

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

identifiable seasonality present

M1:   0.111  M2:   0.028  M3:   0.079  M4:   0.909
M5:   0.200  M6:   0.076  M7:   0.105  M8:   0.294
M9:   0.220  M10:  0.477  M11:  0.470
Q =  0.22
Q(without M2) =  0.24
```

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

identifiable seasonality present

M1:   0.641  M2:   0.040  M3:   0.000  M4:   0.537
M5:   0.200  M6:   0.337  M7:   0.513  M8:   1.897
M9:   0.306  M10:  0.979  M11:  0.964
Q =  0.51
Q(without M2) =  0.57

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

identifiable seasonality present

M1:   1.540  M2:   0.058  M3:   0.455  M4:   1.156
M5:   0.359  M6:   0.282  M7:   0.759  M8:   1.461
M9:   0.708  M10:  1.748  M11:  1.566
Q =  0.79
Q(without M2) =  0.88

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

identifiable seasonality present

M1:   0.018  M2:   0.014  M3:   0.000  M4:   1.032
M5:   0.200  M6:   0.778  M7:   0.137  M8:   0.385
M9:   0.266  M10:  0.450  M11:  0.435
Q =  0.24
Q(without M2) =  0.27

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

identifiable seasonality present

M1:   0.044  M2:   0.010  M3:   0.000  M4:   0.413
M5:   0.200  M6:   0.775  M7:   0.147  M8:   0.478
M9:   0.251  M10:  0.498  M11:  0.498
Q =  0.20
Q(without M2) =  0.22

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

identifiable seasonality present

M1:   0.052  M2:   0.017  M3:   0.000  M4:   0.661
M5:   0.200  M6:   0.735  M7:   0.157  M8:   0.679
M9:   0.384  M10:  0.646  M11:  0.375
Q =  0.25
Q(without M2) =  0.28

*** LUXEMBOURG ***
```

```
Model: (0 1 0)(0 1 1)

identifiable seasonality present

M1:   0.239  M2:   0.087  M3:   0.008  M4:   0.413
M5:   0.200  M6:   0.079  M7:   0.217  M8:   0.521
M9:   0.318  M10:  0.781  M11:  0.781
Q =  0.26
Q(without M2) =  0.28

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

identifiable seasonality present

M1:   0.147  M2:   0.109  M3:   0.206  M4:   0.413
M5:   0.200  M6:   0.009  M7:   0.161  M8:   0.266
M9:   0.222  M10:  0.308  M11:  0.276
Q =  0.19
Q(without M2) =  0.20

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

identifiable seasonality present

M1:   0.013  M2:   0.009  M3:   0.000  M4:   0.413
M5:   0.200  M6:   0.415  M7:   0.100  M8:   0.319
M9:   0.211  M10:  0.231  M11:  0.063
Q =  0.14
Q(without M2) =  0.16

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

identifiable seasonality present

M1:   0.378  M2:   0.574  M3:   0.000  M4:   0.909
M5:   0.200  M6:   0.216  M7:   0.338  M8:   1.055
M9:   0.508  M10:  0.807  M11:  0.366
Q =  0.46
Q(without M2) =  0.44

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

identifiable seasonality present

M1:   0.205  M2:   0.066  M3:   0.005  M4:   0.909
M5:   0.200  M6:   0.691  M7:   0.157  M8:   0.270
M9:   0.117  M10:  0.197  M11:  0.098
Q =  0.26
Q(without M2) =  0.28

*** PORTUGAL ***
Model: (2 1 0)(0 1 1)
```

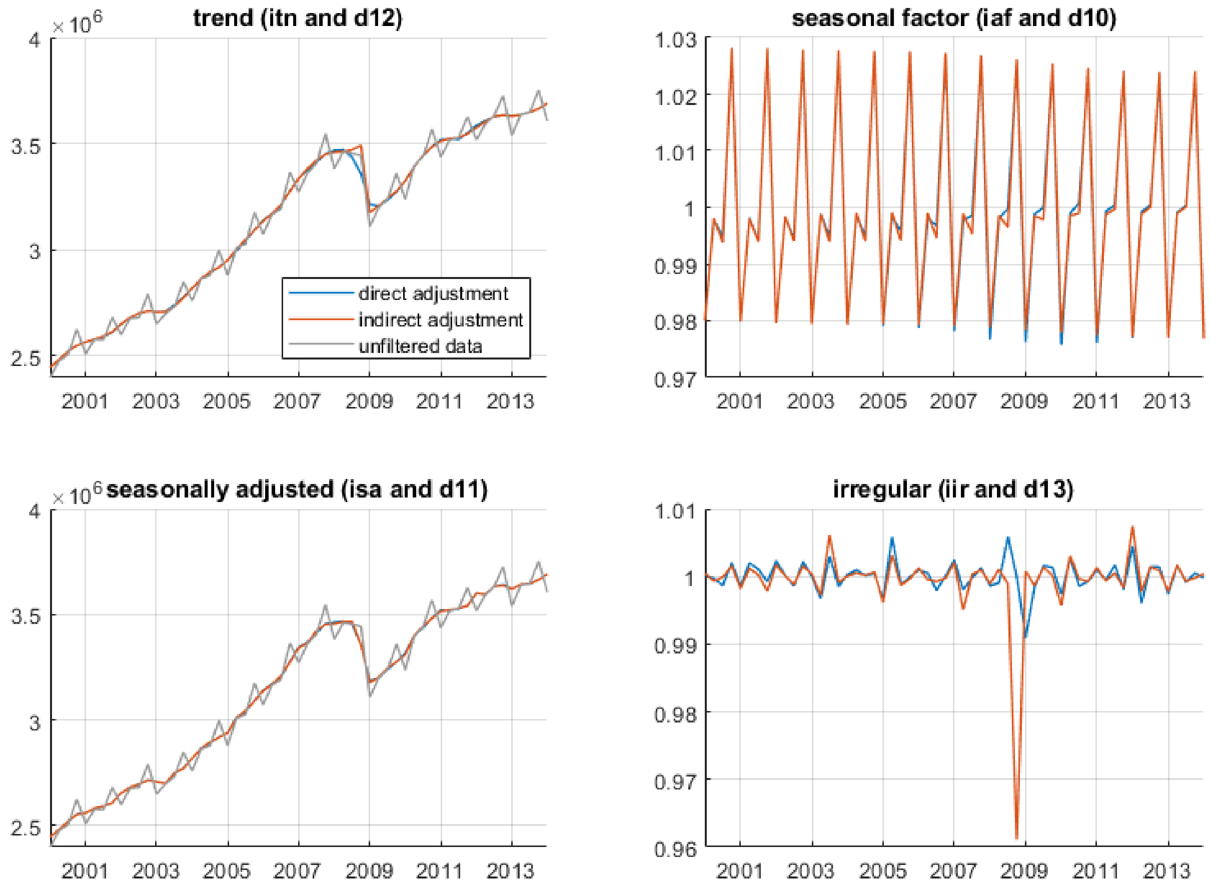
```
identifiable seasonality present

M1:   0.153  M2:   0.062  M3:   0.303  M4:   1.156
M5:   0.200  M6:   0.364  M7:   0.105  M8:   0.319
M9:   0.098  M10:  0.665  M11:  0.665
Q =  0.30
Q(without M2) =  0.33

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

identifiable seasonality present

M1:   0.040  M2:   0.017  M3:   0.002  M4:   0.661
M5:   0.200  M6:   0.114  M7:   0.048  M8:   0.132
M9:   0.021  M10:  0.217  M11:  0.206
Q =  0.13
Q(without M2) =  0.14

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

identifiable seasonality present

M1:   0.171  M2:   0.030  M3:   0.000  M4:   0.413
M5:   0.200  M6:   0.570  M7:   0.170  M8:   0.677
M9:   0.367  M10:  0.482  M11:  0.378
Q =  0.24
Q(without M2) =  0.27

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

identifiable seasonality present

M1:   0.044  M2:   0.008  M3:   0.000  M4:   0.661
M5:   0.200  M6:   0.621  M7:   0.146  M8:   0.396
M9:   0.318  M10:  0.177  M11:  0.139
Q =  0.19
Q(without M2) =  0.21

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

identifiable seasonality present

M1:   0.007  M2:   0.001  M3:   0.000  M4:   0.289
M5:   0.200  M6:   0.688  M7:   0.098  M8:   0.320
M9:   0.109  M10:  0.340  M11:  0.276
Q =  0.13
Q(without M2) =  0.15

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

```
identifiable seasonality present

M1:   0.085  M2:   0.054  M3:   0.000  M4:   0.537
M5:   0.200  M6:   0.178  M7:   0.186  M8:   0.375
M9:   0.273  M10:  0.337  M11:  0.300
Q =  0.20
Q(without M2) =  0.22

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

identifiable seasonality present

M1:   0.596  M2:   0.021  M3:   0.000  M4:   1.280
M5:   0.200  M6:   0.095  M7:   0.683  M8:   1.566
M9:   0.455  M10:  1.322  M11:  1.174
Q =  0.56
Q(without M2) =  0.63

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

identifiable seasonality present

M1:   0.590  M2:   0.185  M3:   0.201  M4:   0.785
M5:   0.200  M6:   0.475  M7:   0.291  M8:   0.604
M9:   0.305  M10:  0.861  M11:  0.861
Q =  0.42
Q(without M2) =  0.45

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

identifiable seasonality present

M1:   1.066  M2:   0.069  M3:   0.131  M4:   0.413
M5:   0.200  M6:   0.580  M7:   0.603  M8:   1.169
M9:   0.324  M10:  1.370  M11:  1.196
Q =  0.56
Q(without M2) =  0.62

-----------------------------------------------------------------------------
Most M-statistics pass, with some exceptions. The M4 and M8 statistic
marginally fail (>1) for some countries, but these are not the most crucial
quality control statistics, so we ignore that. The M1-statistic marginally
fails for the United Kingdom. We ignore this as well, but you are welcome
to look for improvements. Interestingly, TRAMO identifies an ARIMA model
with no seasonal component for UK (model (0 1 1)), yet stable seasonality
is present according to the tests reported in table d8a.
-----------------------------------------------------------------------------
```

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 from FIGURE 1: The differences between direct and ', ...
    'indirect adjustments are small --- except for the spike in the ', ...
    'indirect irregular component. The directly and indirectly seasonally ', ...
    'adjusted series look quite similar, though, and it is not obvious ', ...
    'which one is better. The sum of the individual irregulars in the GFC ', ...
    'is much greater than the irregular identified in the aggregate.']);
```

```
CONCLUSION from FIGURE 1: The differences between direct and indirect
adjustments are small --- except for the spike in the indirect irregular
component. The directly and indirectly seasonally adjusted series look quite
similar, though, and it is not obvious which one is better. The sum of the
individual irregulars in the GFC is much greater than the irregular
indentified in the aggregate.
```

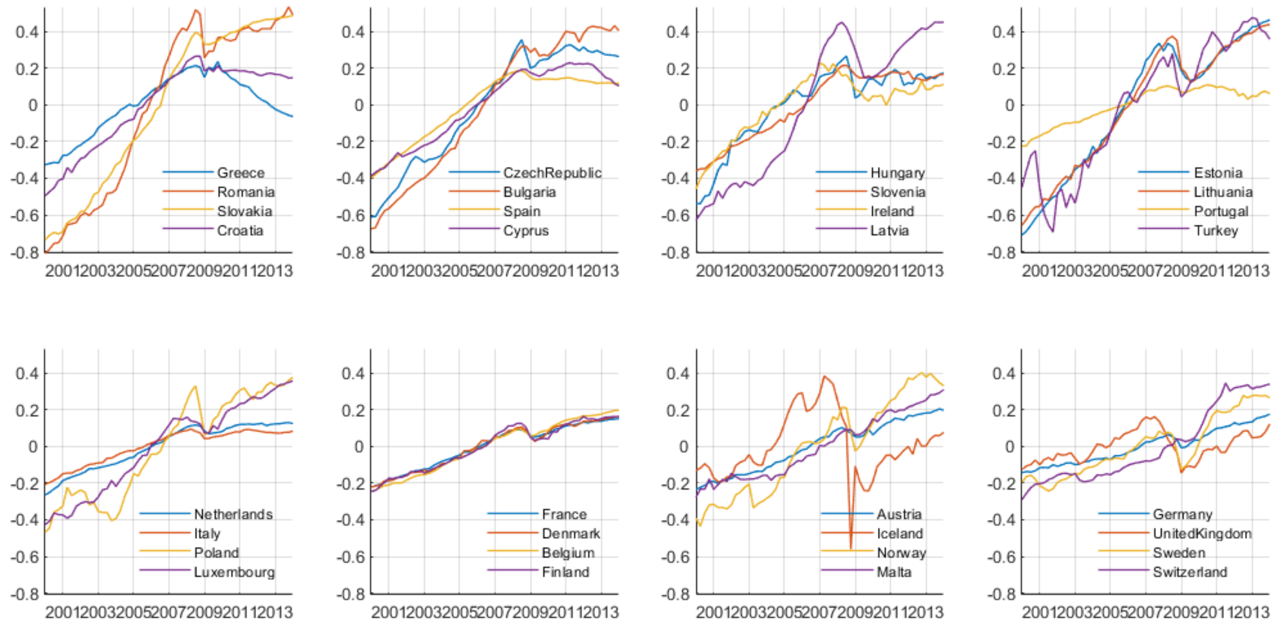


PLOT NORMALIZED LOG GDP FOR ALL COUNTRIES

```
% get list of countries
prop = xgdp.listofseries;
remove = ismember(prop,aggr);
prop(remove) = [];

% sort according to decline of trend growth rate before and after crisis
s = nan(1,numel(prop));
for c = 1:numel(prop)
    loggdp = log(xgdp.(prop{c}).d11.d11);
    y = loggdp(1:35);
    x = [1:numel(y); ones(1,numel(y))]';
    slopebefore = x\y;
    y = loggdp(37:end);
    x = [1:numel(y); ones(1,numel(y))]';
    slopeafter = x\y;
    s(c) = slopeafter(1) - slopebefore(1);
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 from FIGURE 2: 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.']);
```

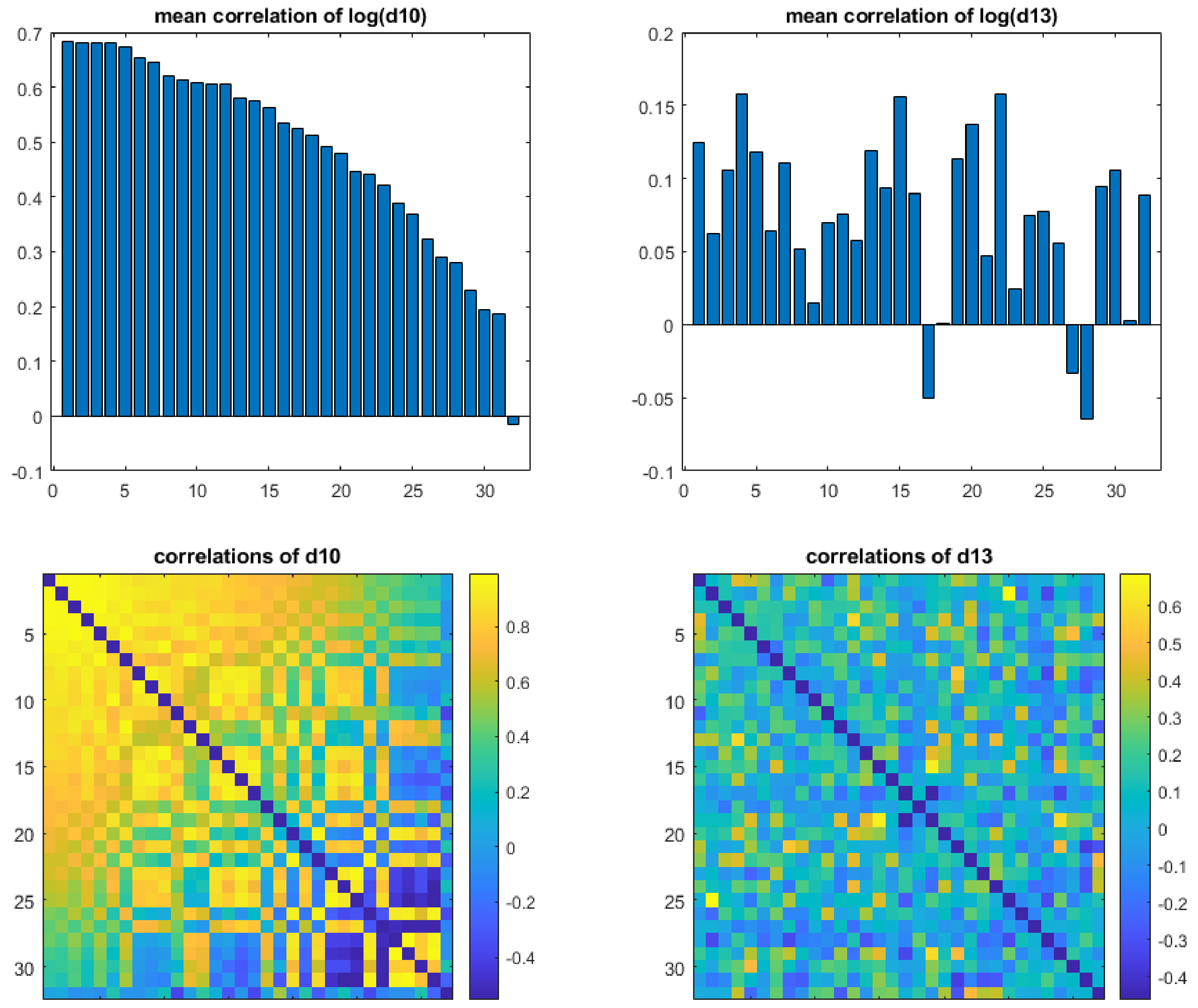
```
CONCLUSION from FIGURE 2: The normalized log seasonally adjusted levels give
a clear view of how the financial / govt debt crisis has affected different
countries. Some were hit brutally but have recovered quickly (e.g. Sweden).
Others are back on their pre-crisis growth rates, but their levels seem 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

```matlab
% 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 from FIGURE 4: Some countries, the UK 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 two groups of countries that behave similarly. ', ...
    'To find out more, we try to cluster them.']);
```

CONCLUSION from FIGURE 4: Some countries, the UK 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 two groups of countries that behave similarly. To find out more, we try to cluster them.

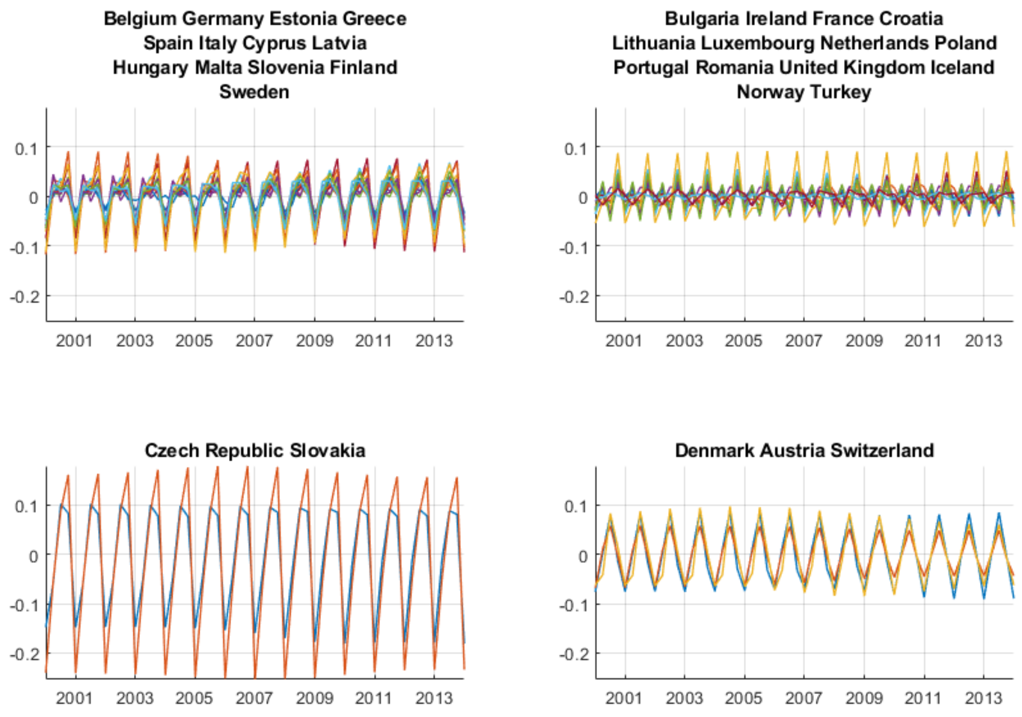


UnitedKingdom (country #32) seems to be much less synchronous than everyone else.

CLUSTER d10

```
% compute kmeans clusters
ncluster = 4;   % number of clusters; you can play around with this
rng default;    % reproducibility
[idx,sa] = kmeans(d10',ncluster,'Distance','sqeuclidean');

% 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 from FIGURE 5: This graph shows the clustering of the ', ...
    'seasonal factors. The Czeck Republic and Slovakia are a clear ', ...
    'cluster. So are Denmark, Austria, and Switzerland. The remaining ', ...
    'countries are divided into two equally large groups, and the UK''s ', ...
    'seasonal pattern appears --- surprisingly --- not that special.']);
```

CONCLUSION from FIGURE 5: This graph shows the clustering of the seasonal factors. The Czeck Republic and Slovakia are a clear cluster. So are Denmark, Austria, and Switzerland. The remaining countries are divided into two equally large groups, and the UK's seasonal pattern appears --- surprisingly --- not that special.



finish up

```
disp(dline);

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

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

Published with MATLAB® R2020a