seasonal: R interface to X-13ARIMA-SEATS

Christoph Sax

February 5, 2015

1 Introduction

seasonal is an easy-to-use and full-featured R-interface to X-13ARIMA-SEATS, the newest seasonal adjustment software developed by the United States Census Bureau. X-13ARIMA-SEATS combines and extends the capabilities of the older X-12ARIMA (developed by the Census Bureau) and TRAMO-SEATS (developed by the Bank of Spain).

If you are new to seasonal adjustment or X-13ARIMA-SEATS, the automated procedures of *seasonal* allow you to quickly produce good seasonal adjustments of time series. Start with the Installation and Getting started section and skip the rest. Alternatively, `demo(seas)` gives an overview of the package functionality.

If you are familiar with X-13ARIMA-SEATS, you may benefit from the flexible input and output structure of *seasonal*. The package allows you to use (almost) all commands of X-13ARIMA-SEATS, and it can import (almost) all output generated by X-13ARIMA-SEATS. The only exception is the 'composite' spec, which is easy to replicate in basic R. Read the Input and Output sections and have a look at the wiki, where the examples from the official X-13ARIMA-SEATS manual are reproduced in R.

2 Installation

2.1 Getting seasonal

The stable version is available from CRAN:

```
install.packages("seasonal")
```

2.2 Getting X-13

seasonal does not include the binary executables of X-13ARIMA-SEATS. They can be obtained precompiled from here (Windows: `x13ashtmlall.zip`). There are guides for building it from source for Ubuntu or Mac OS-X.

Download the file, unzip it and copy `x13ashtml.exe` (or `x13ashtml`, on a Unix system) to any desired location on your file system.

2.3 Telling R where to find X-13

Next, you need to tell *seasonal* where to find the binary executables of X-13ARIMA-SEATS, by setting the specific environmental variable `X13_PATH`. This may be done during your active session in R:

```
Sys.setenv(X13_PATH = "YOUR_X13_DIRECTORY")
```

Exchange `YOUR_X13_DIRECTORY` with the path to your installation of X-13ARIMA-SEATS. Note that the Windows path `C:\something\somemore` has to be entered UNIX-like `C:/something/somemore` or `C:\\something\\somemore`. You can always check your installation with:

```
checkX13()
```

If it works, you may want to set the environmental variable permanently, by adding the `Sys.setenv` line to one of your `.Rprofile` files. The easiest is to use the one located in your home directory, which can be written directly from R:

```
write('Sys.setenv(X13_PATH = "YOUR_X13_DIRECTORY")',
      file = "~/.Rprofile", append = TRUE)
```

If the file does not exist (by default), it will be created. Make sure that you get the quotes right: double quotes around your directory, single quotes around the whole `Sys.setenv` line, such that R understands your string. Check first that the the `Sys.setenv` line works correctly; once it is written you may have to edit `.Rprofile` manually. (Or add a second, overwriting line to it.) For other ways to set an environmental variable permanently in R, see `?Startup`.

3 Getting started

seas is the core function of the *seasonal* package. By default, seas calls the automatic procedures of X-13ARIMA-SEATS to perform a seasonal adjustment that works well in most circumstances:

```
m <- seas(AirPassengers)
```

The first argument of `seas` has to be a time series of class `"ts"`. The function returns an object of class `"seas"` that contains all necessary information on the adjustment.

There are several functions and methods for `"seas"` objects: The `final` function returns the adjusted series, the `plot` method shows a plot with the unadjusted and the adjusted series. The `summary` method allows you to display an overview of the model:

```
final(m)
plot(m)
summary(m)
```

By default, `seas` calls the SEATS adjustment procedure. If you prefer the X11 adjustment procedure, use the following option (see the Input section for details on how to use arbitrary options with X-13):

```
seas(AirPassengers, x11 = "")
```

A default call to `seas` also invokes the following automatic procedures of X -13ARIMA-SEATS:

- Transformation selection (log / no log)
- Detection of trading day and Easter effects
- Outlier detection
- ARIMA model search

Alternatively, all inputs may be entered manually, as in the following example:

```
seas(x = AirPassengers,
     regression.variables = c("td1coef", "easter[1]", "ao1951.May"),
     arima.model = "(0 1 1)(0 1 1)",
     regression.aictest = NULL,
     outlier = NULL,
     transform.function = "log")
```

The `static` command returns the manual call of a model. The call above can be easily generated from the automatic model:

```
static(m)
static(m, coef = TRUE)  # also fixes the coefficients
```

If you have *Shiny* installed, the `inspect` command offers a way to analyze and modify a seasonal adjustment procedure (see the section below for details):

```
inspect(m)
```

4 Input

In *seasonal*, it is possible to use almost the complete syntax of X-13ARIMA- SEATS. This is done via the `...` argument in the `seas` function. The X -13ARIMA-SEATS syntax uses *specs* and *arguments*, with each spec optionally containing some arguments. These spec-argument combinations can be added to `seas` by separating the spec and the argument by a dot (`.`). For example, in order to set the 'variables' argument of the 'regression' spec equal to `td` and `ao1999.jan`, the input to `seas` looks like this:

```
m <- seas(AirPassengers, regression.variables = c("td", "ao1955.jan"))
```

Note that R vectors may be used as an input. If a spec is added without any arguments, the spec should be set equal to an empty string (or, alternatively, to an empty list, as in previous versions). Several defaults of `seas` are empty strings, such as the default `seats = ""`. See the help page (`?seas`) for more details on the defaults. Note the difference between `""` (meaning the spec is enabled but has no arguments) and `NULL` (meaning the spec is disabled).

It is possible to manipulate almost all inputs to X-13ARIMA-SEATS in this way. The best way to learn about the relationship between the syntax of X-13ARIMA-SEATS and *seasonal* is to study the comprehensive list of examples in the wiki. For instance, example 1 in section 7.1 from the manual,

```
series { title  =  "Quarterly Grape Harvest" start = 1950.1
       period =  4
       data   = (8997 9401 ... 11346) }
arima { model = (0 1 1) }
estimate { }
```

translates to R in the following way:

```
seas(AirPassengers,
     x11 = ""),
     arima.model = "(0 1 1)"
)
```

`seas` takes care of the 'series' spec, and no input beside the time series has to be provided. As `seas` uses the SEATS procedure by default, the use of X11 has to be specified manually. When the 'x11' spec is added as an input (like above), the mutually exclusive and default 'seats' spec is automatically disabled. With `arima.model`, an additional spec-argument is added to the input of X-13ARIMA-SEATS. As the spec cannot be used in the same call as the 'automdl' spec, the latter is automatically disabled.

There are some mutually exclusive specs in X-13ARIMA-SEATS. If more than one mutually exclusive spec is included in `seas`, specs are overwritten according the following priority rules:

- Model selection
    1. `arima`
    2. `pickmdl`
    3. `automdl` (default)
- Adjustment procedure
    1. `x11`
    2. `seats` (default)

As an alternative to the `...` argument, spec-arguments can also be supplied as a named list (experimental feature, inspired by base R `save`). This is useful for programming:

```
seas(list = list(x = AirPassengers, x11 = ""))
```

5 Output

seasonal has a flexible mechanism to read data from X-13ARIMA-SEATS. With the `series` function, it is possible to import almost all output that can be generated by X-13ARIMA-SEATS. For example, the following command returns the forecasts of the ARIMA model as a `"ts"` time series:

```
m <- seas(AirPassengers)
series(m, "forecast.forecasts")
```

Because the `forecast.save = "forecasts"` argument has not been specified in the model call, `series` re-evaluates the call with the 'forecast' spec enabled. It is also possible to return more than one output table at the same time:

```
series(m, c("forecast.forecasts", "d1"))
```

You can use either the unique short names of X-13 (such as `d1`), or the the long names (such as `forecasts`). Because the long table names are not unique, they need to be combined with the spec name (`forecast`). See `?series` for a complete list of options.

Note that re-evaluation doubles the overall computation time. If you want to speed it up, you have to be explicit about the output in the model call:

```
m <- seas(AirPassengers, forecast.save = "forecasts")
series(m, "forecast.forecasts")
```

Some specs, like 'slidingspans' and 'history', are time consuming. Re-evaluation allows you to separate these specs from the basic model call:

```
m <- seas(AirPassengers)
series(m, "history.saestimates")
series(m, "slidingspans.sfspans")
```

If you are using the HTML version of X-13, the `out` function shows the content of the main output in the browser:

```
out(m)
```

6 Graphs

There are several graphical tools to analyze a `seas` model. The main plot function draws the seasonally adjusted and unadjusted series, as well as the outliers. Optionally, it also draws the trend of the seasonal decomposition:

```
m <- seas(AirPassengers, regression.aictest = c("td", "easter"))
plot(m)
plot(m, outliers = FALSE)
plot(m, trend = TRUE)
```

The `monthplot` function allows for a monthwise plot (or quarterwise, with the same function name) of the seasonal and the SI component:

```
monthplot(m)
monthplot(m, choice = "irregular")
```

Also, many standard R function can be used to analyze a `"seas"` model:

```
pacf(resid(m))
spectrum(diff(resid(m)))
plot(density(resid(m)))
qqnorm(resid(m))
```

The `identify` method can be used to select or deselect outliers by point and click. Click several times to loop through different outlier types.

```
identify(m)
```

7 Inspect tool

The `inspect` function is a graphical tool for choosing a seasonal adjustment model, using *Shiny*. To install the latest version of Shiny, type:

```
install.packages("shiny")
```

The goal of `inspect` is to summarize all relevant options, plots and statistics that should be usually considered. `inspect` uses a `"seas"` object as its only argument:

```
inspect(m)
```

The `inspect` function opens an interactive window that allows for the manipulation of a number of arguments. It offers several views to analyze the series graphically. With each change, the adjustment process and the visualizations are recalculated. Summary statistics are shown in the first tab. The last tab offers access to all series that can be produced with X-13. The views in `inspect` are also customizable, see the examples in `?inspect`.

8 Chinese New Year, Indian Diwali and other customized holidays

seasonal includes `genhol`, a function that makes it easy to model user-defined holiday regression effects. `genhol` is an R replacement for the equally named software by the Census Office; no additional installation is required. The function uses an object of class `"Date"` as its first argument, which specifies the occurrence of the holiday.

In order to adjust Indian industrial production for Diwali effects, use, e.g.,:

```
data(seasonal) # Indian industrial production: iip
data(holiday)  # dates of Chinese New Year, Indian Diwali and Easter

seas(iip,
x11 = "",
xreg = genhol(diwali, start = 0, end = 0, center = "calendar"),
regression.usertype = "holiday"
)
```

For more examples, including Chinese New Year and complex pre- and post-holiday adjustments, see `?genhol`.

9 Production use

While *seasonal* offers a quick way to adjust a time series in R, it is equally suited for the recurring processing of potentially large numbers of time series. There are two kind of seasonal adjustments in production use:

1. a periodic application of an adjustment model to a time series
2. an automated adjustment to a large number of time series

This section shows how both tasks can be accomplished with *seasonal* and basic R.

9.1 Storing calls and batch processing

`seas` calls are R objects of the standard class `"call"`. Like any R object, calls can be stored in a list. In order to extract the call of a `"seas"` object, you can access the `$call` element or extract the static call with `static()`. For example,

```
# two different models for two different time series
m1 <- seas(fdeaths, x11 = "")
m2 <- seas(mdeaths, x11 = "")

l <- list()
l$c1 <- static(m1)  # static call (with automated procedures substituted)
l$c2 <- m2$call     # original call
```

The list can be stored and re-evaluated if new data becomes available:

```
ll <- lapply(l, eval)
```

which returns another list containing the re-evaluated `"seas"` objects. If you want to extract the final series, use:

```
do.call(cbind, lapply(ll, final))
```

Of course, you also can extract any other series, e.g.:

```
# seasonal component of an X11 adjustment, see ?series
do.call(cbind, lapply(ll, series, "d10"))
```

9.2 Automated adjustment of multiple series

X-13 can also be applied to a large number of series, using automated adjustment methods. This can be accomplished with a loop or an apply function. It is useful to wrap the call to `seas` in a `try` statement; that way, an error will not break the execution. You need to develop an error handling strategy for these cases: You can either drop them, use them without adjustment or switch to a different automated routine.

```
# collect data
dta <- list(fdeaths = fdeaths, mdeaths = mdeaths)

# loop over dta
ll <- lapply(dta, function(e) try(seas(e, x11 = "")))

# list failing models
is.err <- sapply(ll, class) == "try-error"
ll[is.err]

# return final series of successful evaluations
do.call(cbind, lapply(ll[!is.err], final))
```

If you have several cores and want to speed things up, the process is well suited for parallelization (thanks, Matthias Bannert):

```
# a list with 100 time series
largedta <- rep(list(AirPassengers), 100)

library(parallel)  # R-core team, part of R

# set up cluster
cl <- makeCluster(detectCores())

# load 'seasonal' for each node
clusterEvalQ(cl, library(seasonal))

# export data to each node
```

```
clusterExport(cl, varlist = "largedta")

# run in parallel (2.2s on a 8-core Macbook)
parLapply(cl, largedta, function(e) try(seas(e, x11 = "")))

# compare to standard lapply (9.6s)
lapply(largedta, function(e) try(seas(e, x11 = "")))

# finally, stop the cluster
stopCluster(cl)
```

10 License

seasonal is free and open source, licensed under GPL-3. It has been developed for the use at the Swiss State Secretariat of Economic Affairs and is not related to the development of X-13ARIMA-SEATS (license).

Please report bugs and suggestions on Github or send me an e-mail. Thank you!