Bank of Israel **Research Department**

Seasonal adjustment of weekly data by discounted least squares in R[1]

Tim Ginker*

Occasional Paper 2024.03
August 2024

Bank of Israel - http://www.boi.org.il

[1] I'm grateful to Karsten Webel for his review and advice. I would also like to thank Ariel Mantzura, Eyal Argov, Daniel Rosenman, and Ramsis Gara for valuable suggestions and discussions.

* Tim Ginker - Bank of Israel, Information and Statistics Department –
Email: ginker.tim@boi.org.il

חטיבת המחקר, בנק ישראל ת"ד 780 ירושלים 91007
Research Department, Bank of Israel. POB 780, 91007 Jerusalem, Israel

ניכוי עונתיות בתדירות שבועית בשיטת רגרסיה מהוונת ב-R

טים גינקר

תקציר

מאמר זה מציג חבילת boiwsa - R שמיועדת לניכוי עונתיות בתדירות שבועית בשיטת רגרסיה מהוונת. החבילה מספקת ממשק נגיש לביצוע ניכוי עונתיות והיא כוללת פונקציות גמישות ליצירה של משתני התאמה מראש וכן כלים דיאגנוסטיים לבחינת טיב הניכוי. השימוש בחבילה מודגם בשני יישומים אמפיריים: הראשון מבוסס על נתונים מארה״ב של ייצור דלק שמאופיינים ברכיב מגמה משמעותי וקיום מחזור עונתי תוך-שנתי; הדוגמה השנייה מבוססת על נתונים של נרשמים בשירות התעסוקה הישראלי שמאופיינים בשני מחזורי עונתיות הכלולים זה בזה (תוך-שנתי ותוך-חודשי), כמו גם השפעה של שני חגים שאינם במועד קבוע בלוח השנה הלועזי.

מילות מפתח: ניכוי עונתיות; נתונים שבועיים; R.

Seasonal adjustment of weekly data by discounted least squares in R

Tim Ginker

Abstract

This article introduces the R package **boiwsa** for the seasonal adjustment of weekly data based on the discounted least squares method. It provides a user-friendly interface for computing seasonally adjusted estimates of weekly data and includes functions for the creation of country-specific prior adjustment variables, as well as diagnostic tools to assess the quality of the adjustments. The utility of the package is demonstrated through two case studies: one based on US data of gasoline production characterized by a strong trend-cycle and dominant intra-yearly seasonality, and the other based on Israeli data of initial unemployment claims with two seasonal cycles (intra-yearly and intra-monthly) and the impact of two moving holidays.

Keywords: Seasonal Adjustment; Weekly data; R.

1 Introduction

Policymakers and industry practitioners have long utilized weekly and other high-frequency data for timely updates on the state of various sectors of economic activity. Seasonal adjustment is a crucial component of this analysis as it facilitates the economic interpretation of data variations by allowing researchers to analyze these changes separately from the periodic fluctuations introduced by seasonal factors.

This article presents the R package **boiwsa** for seasonal adjustment of weekly data based on the discounted least squares regression (DWR) introduced by Harrison and Johnston (1984). It provides a user-friendly interface for computing seasonally adjusted estimates of weekly data and includes functions for the creation of country-specific prior adjustment variables, as well as diagnostic tools to assess the quality of the adjustments. This equips practitioners with the ability to perform seasonal adjustments on their weekly data, thereby facilitating informed decision-making across a diverse array of applications.

Although attempts to develop procedures for the seasonal adjustment of weekly data date back to the early 20th century (Crum, 1927), the volume of literature addressing this challenge remains relatively limited. Moreover, while there has been notable growth in the number of high-frequency indicators requiring seasonal adjustment, the associated open-source software is less advanced and more intricate than the easily accessible and user-friendly tools commonly available for monthly and quarterly data (Evans et al., 2021)[1].

The seasonal adjustment of high-frequency data presents multiple challenges due to its unique characteristics, which limit the straightforward application of conventional statistical methodologies. Most seasonal adjustment approaches generally encompass two primary steps: detrending and smoothing the cycle sub-series to compute the seasonal components. For instance, with monthly data, an initial application of a wide filter calculates and eliminates the trend. Subsequently, the detrended values are independently smoothed for each month, starting from January and progressing sequentially, to derive the seasonal components.

However, implementing a similar procedure on high-frequency data is not always feasible

[1]Also, see the Seasonality section in the CRAN Task View on Time Series Analysis https://cran.r-project.org/view=TimeSeries, last accessed July 7, 2024.

due to its potential for varying periodicity, presence of multiple seasonal cycles, and the moving window problem. For example, the number of weeks in a year varies between 52 and 53. Additionally, weekly data may exhibit multiple cycles, such as intra-monthly and intra-yearly patterns. As a result, commonly used methods like `X-13 ARIMA-SEATS`, which work well for data with a single and fixed seasonal period (like monthly or quarterly data), cannot be directly applied on weekly data.

To illustrate the challenge associated with the moving window problem, we constructed an artificial unobserved daily series underlying the observed weekly data. This synthetic daily data exhibits a simple monthly seasonal pattern characterized by a linear decrease from 100 to 0 over the course of a month. Consider the first six months of 2023, as depicted in Figure 1. The black line represents the daily data, while, without loss of generality, each blue window corresponds to the first week of a month. Additionally, the green triangles denote the observed weekly averages derived from the black line. It becomes evident that each week aligns with a different portion of the intra-monthly cycle. As a result, even though the underlying daily seasonal pattern is constant, weekly aggregation produces a complex and evolving seasonality. Consequently, in this case, any method assuming a constant number of periods in a seasonal cycle and estimating a seasonal index for each would yield biased estimates.

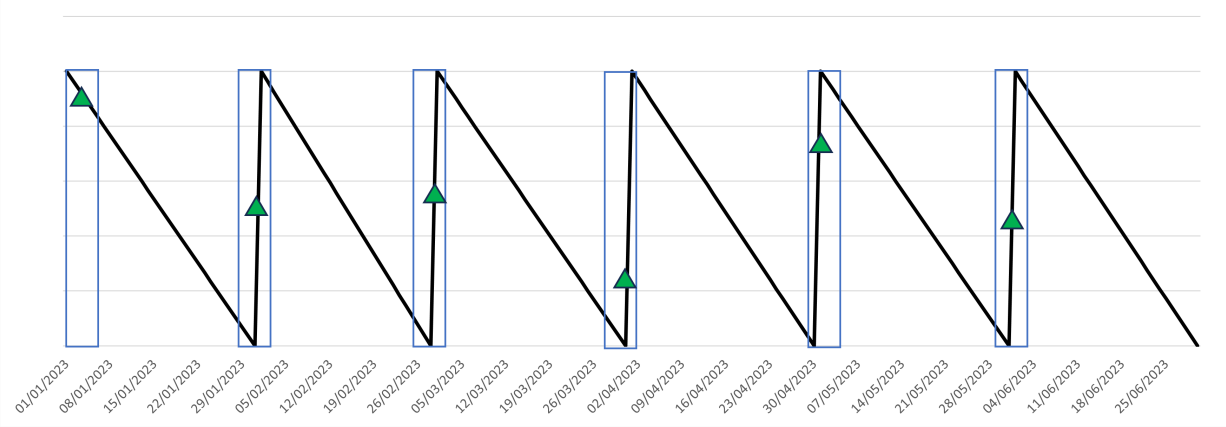

Figure 1: Illustration of the moving window challenge.

Currently, a number of software tools for the seasonal adjustment of weekly data are in various stages of development. One such program is the `MoveReg` weekly seasonal adjust-

ment method developed by the U.S. Bureau of Labor. This program can be executed using EViews or SAS, and its methodology is well documented and straightforward to implement. However, it is not available as open-source code. There are also some promising R packages currently under development, with the most advanced being **Ecce Signum** (McElroy and Livsey, 2022). This package offers a method for the seasonal adjustment of weekly data that is based on the fractional airline model (FAM). However, it is worth mentioning a number of limitations associated with the aforementioned implementation of FAM. First, it implicitly assumes that the data must be differenced to eliminate a trend, which may not necessarily be present in the data. Second, similar to `MoveReg`, it is tailored to handle solely the intra-yearly cycle, a characteristic that could potentially restrict its applicability to data characterized by a substantial intra-monthly cycle, as presented in the application in Section 3.2.

Other software programs, occasionally employed for the seasonal adjustment of weekly data, rely on the Seasonal-Trend decomposition using Loess (STL), developed by Cleveland et al. (1990). A recent extension by Bandara et al. (2021) introduced the concept of Multiple Seasonal-Trend decomposition using Loess (MSTL), thereby extending its application to time series featuring multiple seasonal cycles. It's important to note that both STL and MSTL decompositions are fundamentally based on the cycle subseries smoothing. They assume a constant number of seasonal factors in each cycle, and thus are unable to address the moving window problem discussed earlier. Additionally, these methods lack the capability to effectively account for trading day and moving holiday effects.

Finally, several studies suggest the utilization of forecasting models for the decomposition of time series, such as Prophet (Taylor and Letham, 2018) or TBATS (De Livera et al., 2011). However, due to a different objective function and reliance solely on past observations to compute the components, these methods have been observed to yield less accurate decompositions (Bandara et al., 2021).

The paper is organized as follows. Section 2 outlines our methodology. In Section 3, we provide examples of how to use the **boiwsa** package, which is illustrated through two cases. The first example is based on US gasoline production data, which exhibits a strong trend-cycle and dominant intra-yearly seasonality. The second example is based on Israeli

data of initial unemployment claims, where the level is relatively stable, but there are two seasonal cycles - intra-yearly and intra-monthly - and a pronounced effect of two moving holidays. Section 4 concludes.

2 Methodology

In this section, we describe our methodology. Our approach aligns closely with the locally-weighted least squares procedure introduced by Cleveland et al. (2014) that is implemented in `MoveReg`, albeit with several adjustments. First, instead of relying on differencing to detrend the data, we opt for a more explicit approach by directly estimating the trend through smoothing techniques. Second, we incorporate a variation of DWR to enable the seasonal component to evolve dynamically over time. In contrast with the weight structure proposed by Cleveland et al. (2014), the DWR method employed in this study can be extended so that a different discount factor is applied to each parameter. While, as in Cleveland et al. (2014), the present application uses the same discount factor for all parameters, which is reasonable for weekly data, it could be limiting in future applications on higher frequency data, where it is reasonable to allow each seasonal component to evolve at a different speed.

We consider the following decomposition model of the observed series y_t:

$$y_t = T_t + S_t + H_t + O_t + I_t, \tag{1}$$

where T_t represents the trend component, S_t the seasonal component, H_t the holiday and trading-day effects, O_t and I_t the outlier and irregular components respectively, and t denotes the date of the last day within a given week.

The seasonal component is specified using trigonometric variables as follows

$$\begin{aligned} S_t &= \sum_{k=1}^{K} \left(\alpha_k^y \sin(\frac{2\pi k D_t^y}{n_t^y}) + \beta_k^y \cos(\frac{2\pi k D_t^y}{n_t^y}) \right) + \\ &\quad \sum_{l=1}^{L} \left(\alpha_l^m \sin(\frac{2\pi l D_t^m}{n_t^m}) + \beta_l^m \cos(\frac{2\pi l D_t^m}{n_t^m}) \right), \end{aligned} \tag{2}$$

where D_t^y and D_t^m are the day of the year and the day of the month, and n_t^y and n_t^m are the number of days in the given year or month (Pierce et al., 1984). Thus, the seasonal adjustment procedure takes into account the existence of two cycles, namely intra-yearly and intra-monthly.

It is worth mentioning several differences between working with weekly data as opposed to data with a constant period. A direct relationship exists between the Fourier variable representation in (2) and a basic dummy variable method, which essentially conducts cycle subseries smoothing (Harvey et al., 1997). Consequently, the number of parameters to be estimated remains the same in both techniques. However, when the period is not constant, as in the case of weekly data considered in this paper, applying cycle sub-series smoothing is not feasible. As a result, there is no correspondence between the two representations.

In addition, (2) implicitly assumes the estimation of a daily model. This implies that to incorporate the complete set of trigonometric variables, we would have to set $K = 182$ (analogously to having a complete set of daily dummy variables), which is impractical for most applications. Fortunately, including only a subset of these variables is sufficient for most practical scenarios, usually around 20 (Cleveland et al., 2014), when dealing with a yearly cycle.

Finally, in many existing methods, such as the Fractional Airline Model or the implementation of the local regression method, the current practice is to apply a single differencing by default. However, this approach may not always be necessary or sufficient. In order to prevent overdifferencing, simplify the adjustment process, and facilitate automation, we opt to extract the trend component in the current application using Friedman's Super-Smoother, implemented through the `stats::supsmu()` function (R Core Team, 2013).

Similarly to the X-11 method (Ladiray and Quenneville, 2001), our procedure employs an iterative approach to estimate the different components. The seasonal adjustment algorithm comprises eight steps, which are detailed below:

1. Estimation of trend ($T_t^{(1)}$) using `stats::supsmu()`.

2. Estimation of the Seasonal-Irregular component:

$$y_t - T_t^{(1)} = S_t + H_t + O_t + I_t.$$

2* (Optional) Searching for additive outliers using the method proposed by Findley et al. (1998).

2** (Optional) Identifying the optimal number of Fourier variables.

3. Calculation of seasonal factors, along with other potential factors such as H_t or O_t, is done through DWR on the seasonal-irregular component extracted in Step 2. In this application, the discounting rate decays over the years. For each year t and the observed year τ, a geometrically decaying weight function is represented as: $w_t = r^{|t-\tau|}$, where $r \in (0, 1]$. Several important points are worth mentioning. First, when $r = 1$, the method simplifies to ordinary least squares regression with constant seasonality. On the contrary, smaller values of r permit a more rapid rate of change in the seasonal component. However, it is advised against setting it below 0.5 to prevent overfitting. In addition, the choice of r affects the strength of revisions in the seasonally adjusted data, with higher values of r leading to potentially stronger revisions. Second, our methodology differs from the conventional one-way discounting, enabling the inclusion of future observations in the computation of seasonal factors. This approach circumvents the limitations of the forecasting methods discussed in Bandara et al. (2021). Finally, the choice of year-based discounting is driven by the fact that in traditional discount-weighted regression, even with a conservative choice of $r = 0.95$, in weekly data, observations separated by more than 2 years would carry nearly negligible weight. Therefore, the use of year-based discounting prevents an overly rapid decay which may potentially lead to unstable estimates of the seasonal component.

4. Estimation of the trend ($T_t^{(2)}$) from the seasonally and outlier adjusted series using `stats::supsmu()`.

5. Estimation of the Seasonal-Irregular component:

$$y_t - T_t^{(2)} = S_t + H_t + O_t + I_t$$

6. Computing the final seasonal factors (and possibly other factors such as H_t or O_t) using DWR, as in step 3.

7. Estimation of the final seasonally adjusted series:

$$y_t - S_t - H_t$$

8. Computing the final trend ($T_t^{(3)}$) estimate from the seasonally and outlier adjusted series using `stats::supsmu()`.

3 Use of boiwsa

In this section, we illustrate the use of our package and its key functionalities through two examples. The package is available on GitHub[2], and via the Comprehensive R Archive Network (CRAN)[3]. The first example is based on US data, which exhibits a strong trend-cycle and dominant intra-yearly seasonality. The second example is based on Israeli data, where the level is relatively stable, but there are two seasonal cycles - intra-yearly and intra-monthly - and a pronounced effect of two moving holidays.

3.1 Example 1: Gasoline production in the US

Our first example uses the data on US finished motor gasoline product supplied from the **fpp2** package (Hyndman, 2023), which is presented in Figure 2 below. For convenience, it is stored as a `data.frame` with two columns: one with the series to be adjusted, stored as a numeric vector, and the second with the dates stored as a vector of class `Date`.

[2]See https://github.com/timginker/boiwsa

[3]See https://cran.r-project.org/package=boiwsa

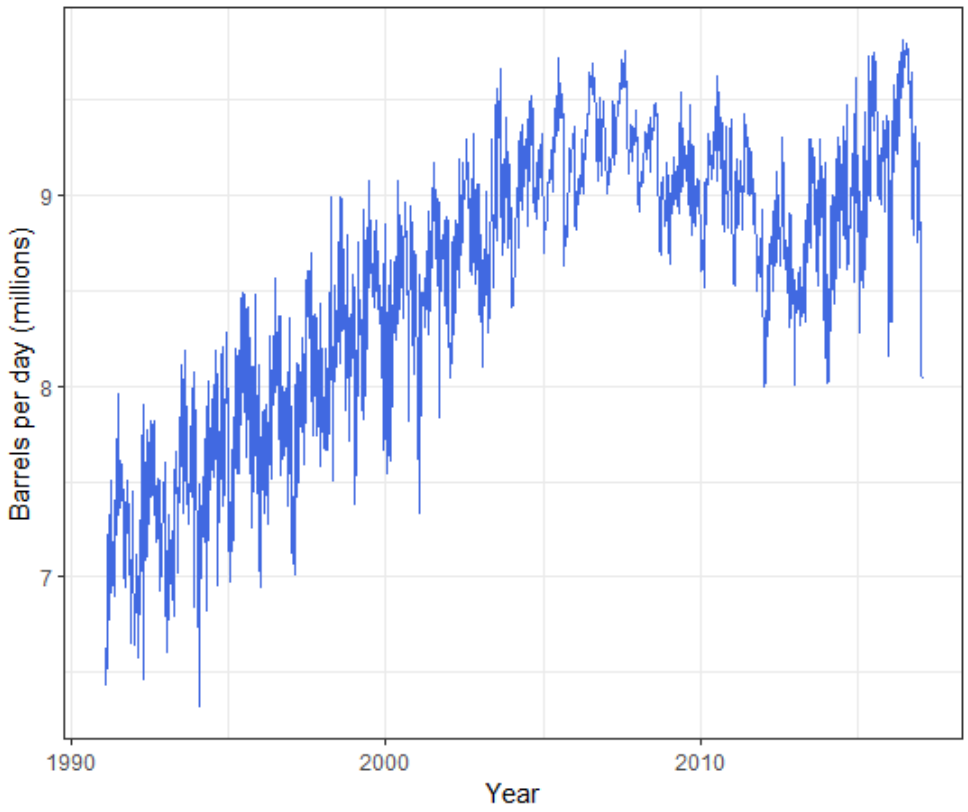

Figure 2: Weekly gasoline production in the US, February 2, 1991 to January 20, 2017. Source: R package fpp2.

Once users have their data loaded, they can use the `boiwsa` function to perform weekly seasonal adjustment. The following code shows the application of the package for seasonal adjustment using automatic model selection. For the full list of arguments and their description, see Table 1 in the Appendix.

```
# loading boiwsa package
library(boiwsa)
# performing seasonal adjustemnt with automatic model selection
res=boiwsa(x=gasoline.data$y,
           dates=gasoline.data$date)
```

In general, the procedure can be applied with minimum interventions and requires only the series to be adjusted and the associated dates. Unless specified otherwise (i.e., `my.k_l`

= NULL), the procedure automatically identifies the best number of trigonometric variables in (2) to capture the yearly (K) and monthly (L) variables based on the Akaike Information Criterion corrected for small sample sizes (AICc). The information criterion is specified by the `ic` option. Like other software, there are three options: "aic", "aicc", and "bic". The weighting decay rate is specified by `r`. By default `r=0.8` which is similar to what is customary in the literature (see Ch.2 in Harvey, 1990).

In addition, the procedure automatically searches for additive outliers (AO) using the method described in Appendix C of Findley et al. (1998). To disable the automatic AO search, set `auto.ao.search = F`. To add user-defined AOs, use the `ao.list` option. As suggested by Findley et al. (1998), the t-statistic threshold for outlier search is by default set to 3.8. However, since high-frequency data are generally more noisy (Proietti and Pedregal, 2023), it could be advantageous to set a higher threshold. This can be specified by the `out.threshold` argument.

The `boiwsa` function returns a list object containing the results (see Table 2 in the Appendix for the full list of values). The seasonally adjusted series is stored in a vector called `sa`. The estimated seasonal factors are stored as `sf`. Note that these contain the seasonal effects represented by the trigonometric variables as well as the user-defined preadjustments in `H`. In addition, the user can see the number of trigonometric terms chosen in automatic search (`my.k_l`); regression coefficients for the last year (`beta`); the position of additive outliers (`ao.list`); the `lm` output of the unweighted OLS regression on the full sample (`m`); final trend estimate (`trend`).

After the seasonal adjustment, we can plot the adjusted data to visualize the seasonal pattern:

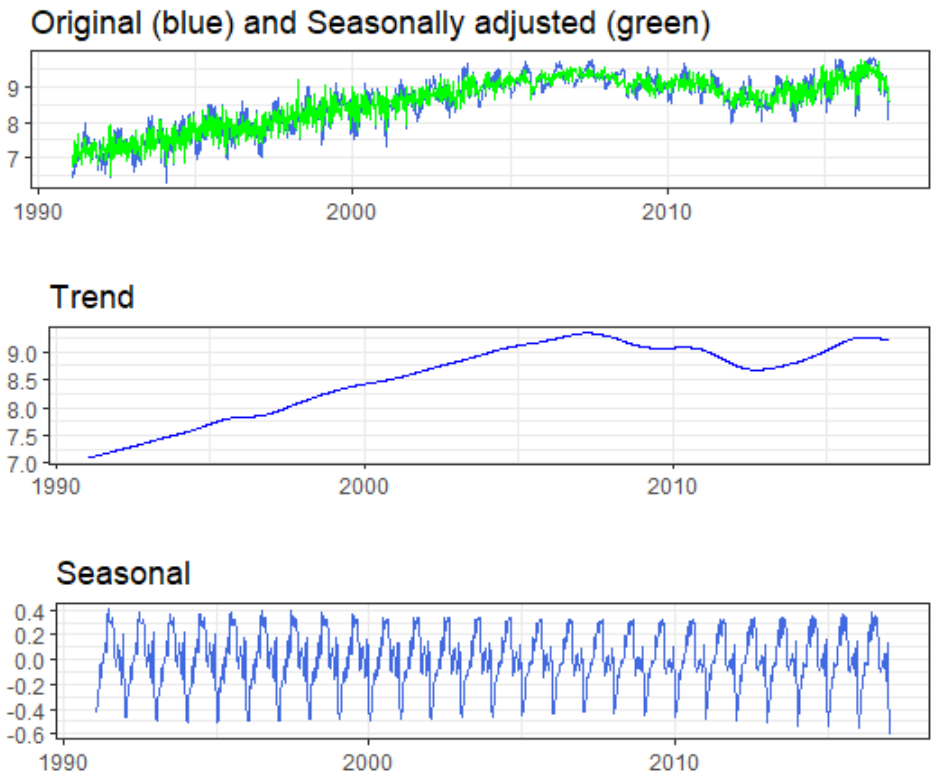

Figure 3: Weekly gasoline production in the US: Original, Seasonally adjusted, and the decomposition.

To compute the autoregressive spectrum, we set the number of lags to 60. In addition, only the first three yearly cycle peaks, and the two monthly peaks are marked by the vertical lines. As we can see from the plot above, generated by `boiwsa::plot_spec`, the series had only an intra-yearly cycle which was completely removed by the adjustment.

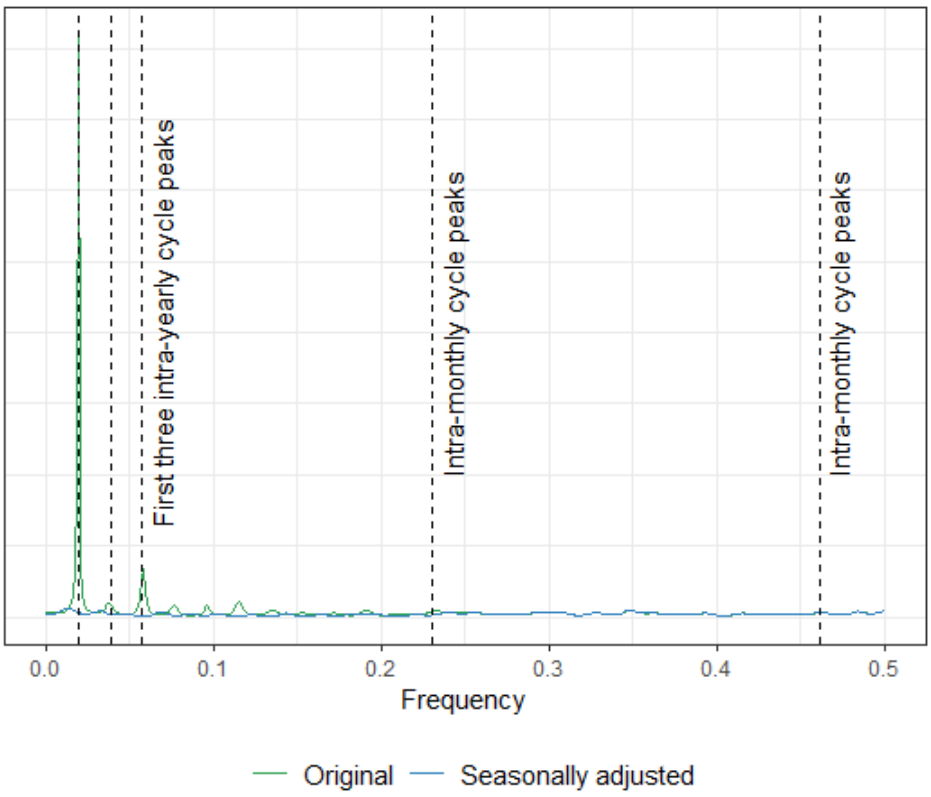

Figure 4: Autoregressive spectrum of weekly gasoline production in the US.

We can also inspect the output to see if the number of trigonometric terms chosen by the automatic procedure matches our visual findings (`res$my.k_1`). The number of yearly terms, K, chosen by the automatic procedure is 12, and the number of monthly terms is zero, which is consistent with the observed spectrum.

3.2 Example 2: Initial unemployment claims in Israel

In this subsection, we present our second example, which is based on Israeli data. The data has no obvious trend but has two pronounced seasonal cycles, intra-yearly and intra-monthly. Moreover, there is a strong impact of two moving holidays and a working day effect.

The series under consideration is the weekly number of initial registrations at the Israeli Employment Service. Due to a prolonged period of structural change associated with the COVID-19 crisis, we limit our sample to January 11, 2014, to January 4, 2020. It is

worth noting that addressing such events in the context of high-frequency data presents a significant challenge. Furthermore, given the duration of the COVID-19 crisis, the standard methods of incorporating outlier variables, which are also available in our package, appear to be incapable of providing a satisfactory solution in this context.

Registration and reporting at the Employment Service are mandatory prerequisites for individuals seeking to receive an unemployment benefit. Therefore, applicants are expected to register promptly after their employment has been terminated. Given that most employment contracts conclude toward the end of the month, an increased number of applications is anticipated at the beginning of the month, leading to an intra-monthly seasonal pattern. Additionally, as can be seen in Figure 5 below, on an annual basis, three distinct peaks are observed, with the final one occurring in August. This peak is closely tied to seasonal workers, leading to the creation of an intra-yearly cycle.

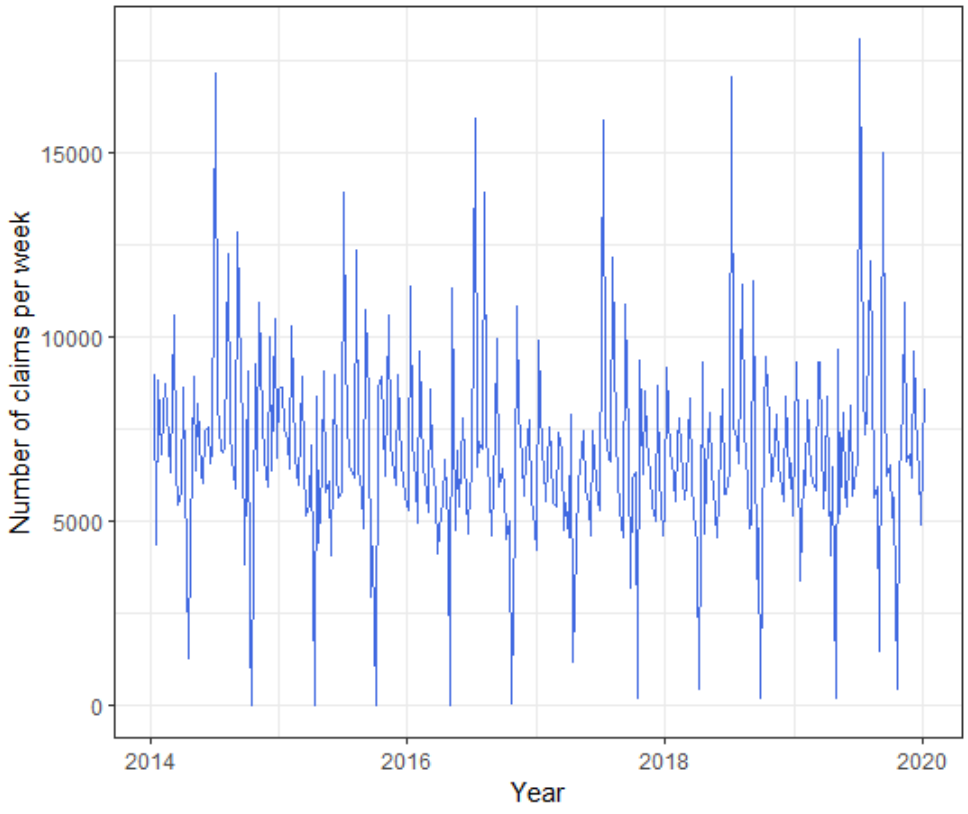

Figure 5: Weekly number of initial unemployment claims in Israel, January 11, 2014 to January 4, 2020. Source: Israeli Employment Service.

Furthermore, each year, there are two weeks in which activity plunges to nearly zero due to the existence of two moving holidays associated with Rosh Hashanah and Passover. Moreover, a working day effect is expected, which leads to a reduced number of applications in weeks with fewer working days. These effects are captured and modeled through additional variables generated by the dedicated functions in **boiwsa**.

To generate a working day variable, we use the `simple_td` function, designed to aggregate the count of full working days within a week and normalize it. This function requires two parameters: the data dates and a `data.frame` object containing information about working days. The `data.frame` should be in a daily frequency and contain two columns: "date" and "WORKING_DAY_PART". For a complete working day, the "WORKING_DAY_PART" column should be assigned a value of 1, for a half working day 0.5, and for a holiday, the value should be set to 0.

Moving holiday variables can be created using the `genhol` function. These variables are computed using the Easter formula in Table 2 of Findley et al. (1998), with the calendar centering to avoid bias, as indicated in the documentation. In the present example, the impact of each holiday is concentrated within a single week, resulting in a noticeable drop and subsequent increase in the number of registrations during the following week. To account for this effect, we employ dummy variables that are globally centered. These dummy variables are created using a custom function named `my_rosh`, which is created for this illustrative scenario.

The code below illustrates the entire process based on the `lbm` dataset: creation of working day adjustment variables using the `simple_td` function; creation of moving holiday variables using the dedicated functions, and adding the combined input into the `boiwsa` function.

```
# creating an input for simple_td
dates_il%>%
  dplyr::select(DATE_VALUE,ISR_WORKING_DAY_PART)%>%
  `colnames<-`(c("date","WORKING_DAY_PART"))%>%
  dplyr::mutate(date=as.Date(date))->df.td
# creating a matrix with a working day variable
```

```
td=simple_td(dates = lbm$date,df.td = df.td)

# generating Rosh Hashanah and Pesach moving holiday variables
rosh=my_rosh(dates=lbm$date,
             holiday.dates = holiday_dates_il$rosh)
# naming (make sure that all the variables in H have distinct names)
colnames(rosh)=paste0("rosh",colnames(rosh))

pesach=my_rosh(dates=lbm$date,
               holiday.dates = holiday_dates_il$pesah,
               start=3,end=-1)
colnames(pesach)=paste0("pesach",colnames(pesach))

# combining our working day and moving holiday variables
H=as.matrix(cbind(rosh[,-1],pesach[,-1],td[,-1]))
# running seasonal adjustment routine
res=boiwsa(x=lbm$IES_IN_W_ADJ,
           dates = lbm$date,
           H=H,
           out.threshold = 3.8)
```

Subsequently, we can visually examine the results of the procedure presented in Figure 6 below.

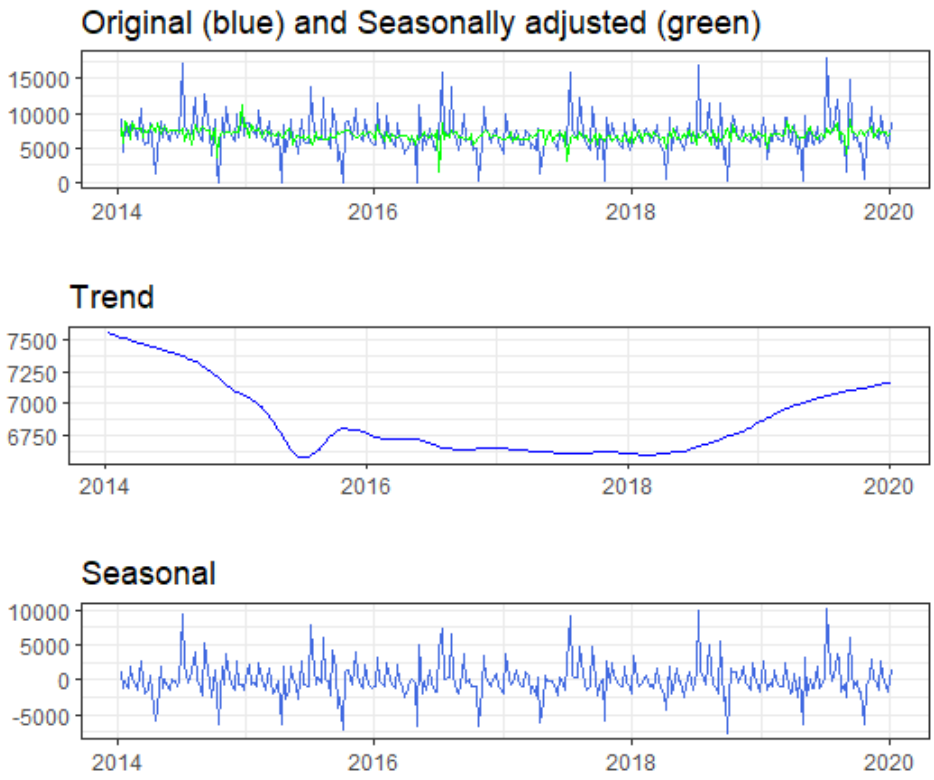

Figure 6: Weekly number of initial unemployment claims in Israel: Original, Seasonally adjusted, and the decomposition with `out.threshold`=3.8.

As we can see in the plot, the procedure has successfully eliminated the annual and monthly seasonal patterns, along with the influences of moving holidays. Nevertheless, it is notable that a few pronounced declines emerge in the seasonally adjusted series from 2016 onward. As previously mentioned, weekly data often contain more noise, potentially prompting the inclusion of outlier variables where they might not be warranted. This inclusion could introduce bias to the seasonal component, consequently leading to the observed distortions.

While the suitability of the approach depends on the specific application, it appears that the recommended outlier threshold of 3.8, typically suggested for monthly data, might be insufficiently conservative for the weekly series. Consequently, a careful examination of the identified outliers is strongly advised. To tackle this issue, one possible solution involves raising the `out.threshold`, as demonstrated in the code below. Figure 7 summarizes the

resulting decomposition.

```
res=boiwsa(x=lbm$IES_IN_W_ADJ,
           dates = lbm$date,
           H=H,
           out.threshold = 5)
```

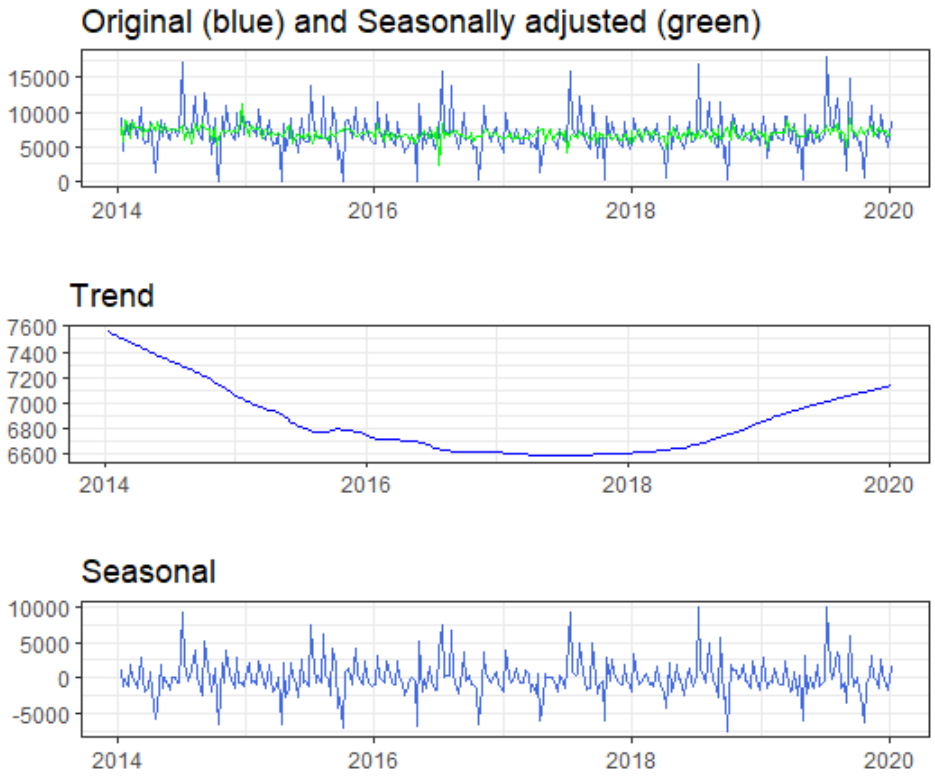

Figure 7: Weekly number of initial unemployment claims in Israel: Original, Seasonally adjusted, and the decomposition with `out.threshold`=5.

Following a thorough visual examination of the seasonally adjusted data, we can now move forward with the spectrum diagnostics. As illustrated in Figure 8 below, corroborating our initial analysis of potential underlying seasonal patterns, it becomes evident that the data has two distinct seasonal cycles. Additionally, it is noteworthy that our procedure successfully removed the corresponding peaks, thereby highlighting its effectiveness.

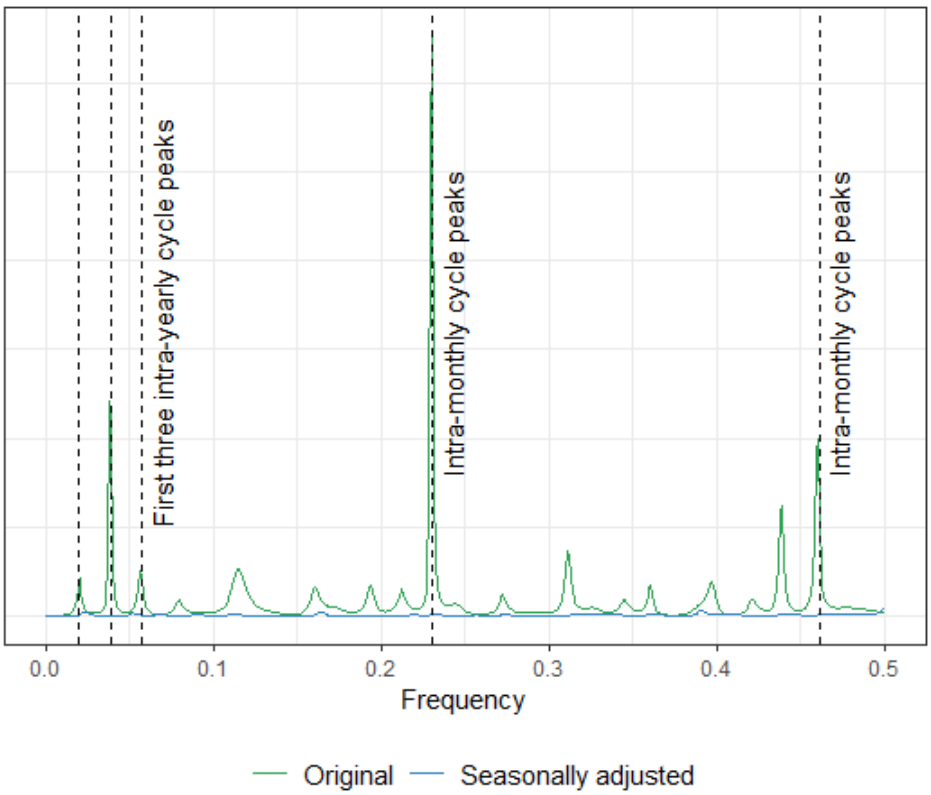

Figure 8: Autoregressive spectrum of a weekly number of initial unemployment claims in Israel.

4 Discussion

The package **boiwsa** is developed to equip practitioners with the ability to perform seasonal adjustments on the weekly data, thereby facilitating informed decision-making across a diverse array of applications. It provides a user-friendly interface for computing seasonally adjusted estimates of weekly data and includes functions for the creation of country-specific prior adjustment variables, as well as diagnostic tools to assess the quality of the adjustments. The empirical applications presented in this paper demonstrate its functionality and ability to perform under various practical scenarios.

References

Bandara, K., R. J. Hyndman, and C. Bergmeir (2021). MSTL: A seasonal-trend decomposition algorithm for time series with multiple seasonal patterns. *arXiv preprint arXiv:2107.13462*.

Cleveland, R. B., W. S. Cleveland, J. E. McRae, and I. Terpenning (1990). STL: A seasonal-trend decomposition. *Journal of Official Statistics 6*(1), 3–73.

Cleveland, W. P., T. D. Evans, and S. Scott (2014). Weekly Seasonal Adjustment - A Locally-weighted Regression Approach. Economic working papers, Bureau of Labor Statistics.

Crum, W. (1927). Weekly fluctuations in outside bank debits. *The Review of Economic Statistics*, 30–36.

De Livera, A. M., R. J. Hyndman, and R. D. Snyder (2011). Forecasting time series with complex seasonal patterns using exponential smoothing. *Journal of the American Statistical Association 106*(496), 1513–1527.

Evans, T. D., B. C. Monsell, and M. Sverchkov (2021). Review of available programs for seasonal adjustment of weekly data. General methodology, Bureau of Labor Statistics.

Findley, D. F., B. C. Monsell, W. R. Bell, M. C. Otto, and B.-C. Chen (1998). New capabilities and methods of the x-12-arima seasonal-adjustment program. *Journal of Business & Economic Statistics 16*(2), 127–152.

Ginker, T. (2024). *boiwsa: Seasonal Adjustment of Weekly Data*. R package version 1.1.0.

Harrison, P. J. and F. R. Johnston (1984). Discount weighted regression. *The Journal of the Operational Research Society 35*(10), 923–932.

Harvey, A., S. J. Koopman, and M. Riani (1997). The modeling and seasonal adjustment of weekly observations. *Journal of Business & Economic Statistics 15*(3), 354–368.

Harvey, A. C. (1990). *Forecasting, structural time series models and the Kalman filter*. Cambridge University Press.

Hyndman, R. (2023). *fpp2: Data for "Forecasting: Principles and Practice" (2nd Edition)*. R package version 2.5.

Ladiray, D. and B. Quenneville (2001). *Seasonal adjustment with the X-11 method*. Springer New York, NY.

McElroy, T. S. and J. A. Livsey (2022). Ecce Signum: An R Package for multivariate signal extraction and time series analysis. *arXiv preprint arXiv:2201.02148*.

Pierce, D. A., M. R. Grupe, and W. P. Cleveland (1984). Seasonal adjustment of the weekly monetary aggregates: A model-based approach. *Journal of Business & Economic Statistics 2*(3), 260–270.

Proietti, T. and D. J. Pedregal (2023). Seasonality in high frequency time series. *Econometrics and Statistics 27*, 62–82.

R Core Team (2013). *R: A Language and Environment for Statistical Computing*. Vienna, Austria: R Foundation for Statistical Computing. ISBN 3-900051-07-0.

Taylor, S. J. and B. Letham (2018). Forecasting at scale. *The American Statistician 72*(1), 37–45.

Appendix

Table 1: Input arguments for **boiwsa**

Input argument	Description
`x`	Numeric vector with series to be seasonally adjusted
`dates`	Vector of class "Date", containing the data dates
`r`	Defines the rate of decay of the weights. Should be between zero and one. By default is set to 0.8
`auto.ao.seacrh`	Boolean. Search for additive outliers
`out.threshold`	t-statistic threshold in outlier search. By default is set to 3.8 as suggested by Findley et al. (1998)
`ao.list`	Vector with user-specified additive outliers in a date format
`my.k_l`	Numeric vector defining the number of yearly and monthly trigonometric variables. If NULL, is found automatically using the information criteria (AICc)
`H`	Matrix with holiday/working day factors or other user-defined preadjustemts
`ic`	Information criterion used in the automatic search for the number of trigonometric regressors. There are three options: aic, aicc, and bic. aicc is set as default
`method`	Decomposition type: additive or multiplicative (log transformation)

Table 2: Output values for **boiwsa**

Output value	Description
`sa`	Numeric vector with seasonally adjusted series
`x`	Numeric vector with series to be seasonally adjusted
`my.k_l`	Number of trigonometric variables used to model the seasonal pattern
`sf`	Estimated seasonal effects. Note that these contain the seasonal effects represented by the trigonometric variables in (2) as well as the user-defined preadjustments in `H`
`hol.factors`	Estimated holiday effects or other user-defined variables supplied in H
`out.factors`	Estimated outlier effects
`beta`	DWR coefficients for the last year
`trend`	Final trend estimate ($T_t^{(3)}$)
`ao.list`	Additive outlier dates
`m`	`lm` object. Unweighted OLS regression on the full sample