Journal of Statistical Software

MMMMMM YYYY, Volume VV, Issue II. *http://www.jstatsoft.org/*

Mixed Frequency Data Sampling Regression Models: the R Package midasr

Eric Ghysels
University of North Carolina

Virmantas Kvedaras
Vilnius University

Vaidotas Zemlys
Vilnius University

Abstract

When modeling economic relationships it is increasingly common to encounter data sampled at different frequencies. We introduce R package **midasr** which enables estimating regression models with variables sampled at different frequencies within a MIDAS regression framework put forward in work by Ghysels, Santa-Clara, and Valkanov (2002). In this article we define a general autoregressive MIDAS regression model with multiple variables of different frequencies and show how it can be specified using familiar R formula interface and estimated using various optimisation methods chosen by the researcher. We discuss how to check the validity of estimated model both in terms of numerical convergence and statistical adequacy of a chosen regression specification, how to do the model selection based on a information criteria, how to assess forecasting accuracy of the MIDAS regression model and do a forecast aggregation of different MIDAS regression models. We illustrate package capabilities using a simulated MIDAS regression model and give two empirical examples of application of MIDAS regression.

Keywords: MIDAS, specification test.

1. Introduction

Regression models involving data sampled at different frequencies are of general interest. In this document we introduce a R package **midasr** for the regression modeling with the mixed frequency data based on a framework put forward in recent work by Ghysels *et al.* (2002), Ghysels, Santa-Clara, and Valkanov (2006a) and Andreou, Ghysels, and Kourtellos (2010) using so called MIDAS, meaning Mi(xed) Da(ta) S(ampling), regressions.

In a general framework of regressions with functional constraints on parameters, the **midasr** package not only provides similar functionality within a standard R framework of the model specification comparable to that available in the usual functions `lm` or `nls`, but also deals

with an extended model specification analysis for MIDAS regressions.

Several recent surveys on the topic of MIDAS are worth mentioning at the outset. They are: Andreou, Ghysels, and Kourtellos (2011) who review more extensively some of the material summarized in this document, Armesto, Engemann, and Owyang (2010) who provide a very simple introduction to MIDAS regressions and finally Ghysels and Valkanov (2012) who discuss volatility models and mixed data sampling.

Econometric analysis of MIDAS regressions appears in Ghysels, Sinko, and Valkanov (2006b), Andreou *et al.* (2010), Bai, Ghysels, and Wright (2012), Kvedaras and Račkauskas (2010), Rodriguez and Puggioni (2010), Wohlrabe (2009), among others.

MIDAS regression can also be viewed as a reduced form representation of the linear projection which emerges from a state space model approach - by reduced form we mean that the MIDAS regression does not require the specification of a full state space system of equations. Bai *et al.* (2012) show that in some cases the MIDAS regression is an exact representation of the Kalman filter, in other cases it involves approximation errors which are typically small. The Kalman filter, while clearly optimal as far as linear projections goes, has several disadvantages (1) it is more prone to specification errors as a full system of measurement and state equations is required and as a consequence (2) requires a lot more parameters, which in turn results in (3) computational complexities which often limit the scope of applications. In contrast, MIDAS regressions - combined with forecast combination schemes if large data sets are involved (see Andreou, Ghysels, and Kourtellos (2013)) are computationally easy to implement and are less prone to specification errors.

The key feature of the package is its flexibility in terms of the model formulation and estimation, which allows for the[1]:

- estimation of regression models with its parameters defined (restricted) by a certain functional constraint using familiar R `formula` interface allowing any choice of a constraint, which can be selected from a standard list or can be customized using user-defined R functions;
- parsimonious aggregation-linked restrictions (as e.g., in Ghysels 2013) as a special case;
- estimation of MIDAS models with many variables and (numerous) different frequencies;
- constrained, partially constrained, or unconstrained estimation of the model;
- various mixtures of restrictions/weighting schemes and also lag orders as they can be specific to each series;
- statistical testing for the adequacy of the model specification and the imposed functional constraint;
- information criteria and testing-based selection of models;
- forecasting and nowcasting functionality, including various forecast combinations.

[1] Ghysels (2013) also developed a package for MATLAB which deals with the estimation and information criteria-based specification of MIDAS regressions as well as forecasting and nowcasting of low frequency series. All of the **midasr** features replicate or extend features provided by the said package. The key extensions are: the specification of any user-defined functional contraint, the inclusion of multiple variables of different frequency and different functional constraints, and the testing of the adequacy of a chosen model specification.

Suppose $\{y_t, t \in \mathbb{Z}\}$ is a univariate process observed at low frequency. Lags of the process are denoted by $By_t = y_{t-1}$, where B is the low frequency lag operator. A MIDAS regression involves linear projections using stochastic processes $\{x_\tau^{(i)}, \tau \in \mathbb{Z}\}$, $i = 0, \ldots, k$, observed at a higher frequency, i.e., for each low frequency period $t = t_0$ we observe the process $x_\tau^{(i)}$ at $m_i \in \mathbb{N}$ high frequency periods $\tau = (t_0 - 1)m_i + 1, \ldots, t_0 m_i$. Throughout the article we represent i-th high frequency period τ in terms of low frequency period t as $\tau = (t-1)m_i + j$, $j = 1, \ldots, m_i$. Note that this notation does not exclude the case $m_i = 1$. In that case the high frequency process $x_\tau^{(i)}$ is observed at the same frequency as the low frequency process y_t. However we require that $m_i \geq 1$, such that the process y_t is observed at the lowest frequency. Lags of the processes $x_\tau^{(i)}$ are denoted by $Lx_\tau^{(i)} = x_{\tau-1}$, where L is the high frequency lag operator, which operates on the lag irrespective of the frequency of the process.

The package deals with any specification of mixed-frequency regression model which can be represented as

$$y_t - \alpha_1 y_{t-1} - \cdots - \alpha_p y_{t-p} = \sum_{i=0}^{k} \sum_{j=0}^{l_i} \beta_j^{(i)} x_{tm_i - j}^{(i)} + \varepsilon_t, \tag{1}$$

where we require

$$E(\varepsilon_t | y_{t-1}, \ldots, y_{t-p}, x_{tm_0}^{(0)}, \ldots, x_{tm_0 - l_i}^{(0)}, \ldots, x_{tm_k}^{(k)}, \ldots, x_{tm_k - l_k}^{(k)}) = 0,$$

so that the equation (1) is identified as a projection equation.

The model stated in the equation (1) can be estimated in the usual time series regression fashion or using a Bayesian approach. However, the number of parameters in this model $d = p + \sum_{i=0}^{k} l_i$ can be very large in terms of the number n of available observations of y_t[2]. Since the estimation of the model can easily become infeasible, whenever either larger differences in frequencies or more variables and/or higher lag orders prevail, Ghysels *et al.* (2002) introduced a sufficiently flexible parametric restriction to be imposed on the original parameters,

$$\beta_j^{(i)} = f_i(\boldsymbol{\gamma}_i, j),\ j = 0, \ldots, l_i,\ \boldsymbol{\gamma}_i = (\gamma_1^{(i)}, \ldots, \gamma_{q_i}^{(i)}),\ q_i \in \mathbb{N}. \tag{2}$$

This approach greatly reduces the number of parameters to be estimated, from d to $q = \sum_{i=0}^{h_i} q_i$, which is assumed to be always considerably less than the number of observations available at the lowest frequency. This gain is offset by the fact that (1) becomes non-linear model, however if the parameters of an underlying data generating process did follow a certain functional constraint which is perfectly or well approximated by a constraint function chosen by a researcher, then significant efficiency gains could be achieved from the imposed constraints. Figure 1 plots the out-of-sample prediction precision (left figure) and the parameter estimation precision (right figure) in an unconstrained and constrained simple model with correct and approximate restrictions (see Appendix A for details).

As can be seen, even an incorrect constraint might be useful whenever the number of degrees of freedom in an unconstrained model is low and, consequently, one cannot rely on the large sample properties of unconstrained estimators. Furthermore, this approach seems to be necessary whenever estimation is simply infeasible because of the lack of degrees of freedom.

[2]In the MIDAS literature it is common to have $k_i \geq m_i$ and m_i can be large, for example monthly versus daily data

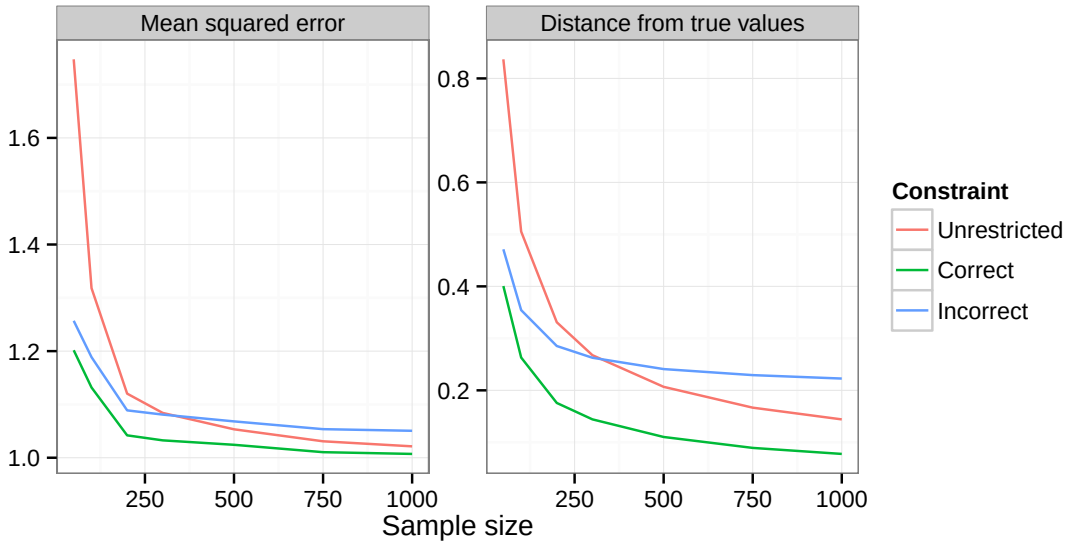

Figure 1: A plot depicting efficiency gains when the correct non-linear constraint is imposed. The left panel plots the average out-of-sample prediction accuracy against sample size. The right panel plots out the average euclidean distance of estimated model parameters to their true values.

2. Theory

The model (1) can be rewritten in a more compact form:

$$\alpha(B)y_t = \boldsymbol{\beta}(L)^\top \boldsymbol{x}_{t,0} + \varepsilon_t, \tag{3}$$

where $\alpha(z) = 1 - \sum_{j=1}^{p} \alpha_j z^j$ and

$$\boldsymbol{x}_{t,0} := \left(x_{tm_0}^{(0)}, \ldots, x_{tm_i}^{(i)}, \ldots, x_{tm_l}^{(l)}\right)^\top,$$

$$\boldsymbol{\beta}(z) = \sum_{j=0}^{l} \boldsymbol{\beta}_j z^j, \ \boldsymbol{\beta}_j = \left(\beta_j^{(0)}, \ldots, \beta_j^{(i)}, \ldots, \beta_j^{(l)}\right)^\top,$$

$$L^j \boldsymbol{x}_{t,0} := \boldsymbol{x}_{t,j} = \left(L^j x_{tm_0}^{(0)}, \ldots, L^j x_{tm_i}^{(i)}, \ldots, L^j x_{tm_l}^{(l)}\right)^\top.$$

In order to simplify notation, without loss of generality, a single order of the lag polynomials is used with l being the maximum lag order. If the orders of some components of $\boldsymbol{\beta}(z)$ are smaller, it is easy to set some coefficients of the polynomial equal to zero.

We require the existence of the continuous second derivative of functional constraint with respect to its parameters i.e., $\frac{\partial^2 f_i}{\partial \boldsymbol{\gamma}_i \partial \boldsymbol{\gamma}_i^\top}$. The functional constraints can vary with each variable and/or frequency, and therefore we use $\boldsymbol{\gamma}$ to represent a vector of all the parameters of a restricted model with $q = \dim(\boldsymbol{\gamma})$ their total number.

As will be shown in the next section, all variants of the usual linear (in terms of variables) MIDAS regression model are covered by regression (3) via the specification of functional

constraints. When each restriction function is an identity mapping, one obtain an unrestricted MIDAS regression model.[3]

2.1. Frequency alignment

It is instructive to rewrite the model (1) in a matrix notation. We start with a few examples. Suppose y_t is observed quarterly and we want to explain its variation with the variable x_τ, which is observed monthly. Since each quarter has three months, the frequency m is 3 in this example. Suppose we assume that the monthly data in the current and the previous quarter has explanatory power. This means that for each quarter t we want to model y_t as a linear combination of variables $x_{3t}, x_{3t-1}, x_{3t-2}$ observed in the quarter t and variables y_{t-1} and $x_{3(t-1)}, x_{3(t-1)-1}, x_{3(t-1)-2}$ observed in the previous quarter $t-1$. In matrix notation the MIDAS model (1) for this example is:

$$\begin{bmatrix} y_2 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} y_1 \\ \vdots \\ y_{n-1} \end{bmatrix} \alpha_1 + \begin{bmatrix} x_6 & \dots & x_1 \\ \vdots & \vdots & \vdots \\ x_{3n} & \dots & x_{3n-5} \end{bmatrix} \begin{bmatrix} \beta_0 \\ \vdots \\ \beta_5 \end{bmatrix} + \begin{bmatrix} \varepsilon_2 \\ \vdots \\ \varepsilon_n \end{bmatrix}$$

By writing the model in the matrix notation we have transformed high-frequency variable x_τ into a low-frequency vector $(x_{3t}, \dots, x_{3t-5})^\top$. We call this transformation the frequency alignment. Note that we require that the number of observations of x_τ is exactly $3n$.

Let us examine another example. Suppose we have another variable z_t observed weekly which we want to add to the model. The model (1) does not allow varying frequency ratios, so we need to assume that each month has exactly 4 weeks. If months do not always have four weeks, as they do in practice, one can simply think of this model as taking a fixed set of weekly lags. The frequency m for the variable z_τ is then 12. We use again the current and previous quarter data for explaining variation in y_t. This means that for quarter t we model y_t as a linear combination of variables $x_{3t}, x_{3t-1}, x_{3t-2}$ and $z_{12t}, z_{12t-1}, \dots, z_{12t-11}$ observed in the quarter t, and variables y_{t-1}, $x_{3(t-1)}, \dots, x_{3(t-1)-2}$ and $z_{12(t-1)}, z_{12(t-1)-1}, \dots, z_{12(t-1)-11}$ observed in the quarter $t-1$. The model in the matrix form is then:

$$\begin{bmatrix} y_2 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} y_1 \\ \vdots \\ y_{n-1} \end{bmatrix} \alpha_1 + \begin{bmatrix} x_6 & \dots & x_1 \\ \vdots & \vdots & \vdots \\ x_{3n} & \dots & x_{3n-5} \end{bmatrix} \begin{bmatrix} \beta_0 \\ \vdots \\ \beta_5 \end{bmatrix} + \begin{bmatrix} z_{24} & \dots & z_1 \\ \vdots & \vdots & \vdots \\ z_{12n} & \dots & z_{12n-23} \end{bmatrix} \begin{bmatrix} \gamma_0 \\ \vdots \\ \gamma_{23} \end{bmatrix} + \begin{bmatrix} \varepsilon_2 \\ \vdots \\ \varepsilon_n \end{bmatrix}$$

In this example we have aligned x_τ into a vector $(x_{3t}, .., x_{3t-5})^\top$ and z_τ into a vector $(z_{12t}, \dots, z_{12t-23})^\top$. Again we require that the number of observations of high frequency variables are multiple of n, with multiplication factor being the corresponding frequencies. This is not a restrictive assumption in practical applications as will be further explained in the section 3.

Let us return to the general case of the model (1). We *align the frequency* of high-frequency variable x_τ by transforming it to the low-frequency vector $(x^{(i)}_{tm_i}, x^{(i)}_{tm_i-1}, \dots, x^{(i)}_{tm_i-l})^\top$. The model (1) is then expressed in the matrix notation as follows:

$$\begin{bmatrix} y_l \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} y_{l-1} & \dots & y_{l-p} \\ \vdots & \vdots & \vdots \\ y_{n-1} & \dots & y_{n-p} \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \vdots \\ \alpha_p \end{bmatrix} + \sum_{i=0}^{k} \boldsymbol{X}^{(i)} \begin{bmatrix} \beta^{(i)}_0 \\ \vdots \\ \beta^{(i)}_l \end{bmatrix} + \begin{bmatrix} \varepsilon_l \\ \vdots \\ \varepsilon_n \end{bmatrix},$$

[3] See Foroni, Marcellino, and Schumacher (2014).

where

$$\boldsymbol{X}^{(i)} := \begin{bmatrix} x^{(i)}_{um_i} & x^{(i)}_{um_i-1} & \dots & x^{(i)}_{um_i-l} \\ x^{(i)}_{(u+1)m_i} & x^{(i)}_{(u+1)m_i-1} & \dots & x^{(i)}_{(u+1)m_i-l} \\ \vdots & \vdots & \dots & \vdots \\ x^{(i)}_{tm_i} & x^{(i)}_{tm_i-1} & \dots & x^{(i)}_{tm_i-l} \\ \vdots & \vdots & \dots & \vdots \\ x^{(i)}_{(n-1)m_i} & x^{(i)}_{(n-1)m_i-1} & \dots & x^{(i)}_{(n-1)m_i-l} \\ x^{(i)}_{nm_i} & x^{(i)}_{nm_i-1} & \dots & x^{(i)}_{nm_i-l} \end{bmatrix}, \tag{4}$$

and u is the smallest integer such that $um_i - l > 0$ and $u > p$.

The purpose of this subsection was to show how the frequency alignment procedure turns a MIDAS regression into a classical time series regression where all the variables are observed at the same frequency.

2.2. Estimation

Equation (3) can be estimated directly via ordinary least squares (OLS), without restrictions on the parameters. This is a so called U-MIDAS regression model, see Foroni *et al.* (2014). Furthermore, a consistent non-parametric approach could be used to estimate the underlying parameters of a function as e.g., in Breitung, Roling, and Elengikal (2013). Since, none of these approaches use a parametric functional constraint, they can be estimated using already available R packages. The **midasr** package aims at the estimation of mixed frequency models with some parametric functional constraints.

While model (3) is a linear model in terms of variables, any non-linear functional constraints will result in non-linearities with respect to the parameters $\boldsymbol{\gamma}$. Therefore, in the general case, we use in the function `midas_r` the non-linear least squares (NLS) estimator of parameters $\boldsymbol{\gamma}$ of a restricted model (3) as defined by

$$\widehat{\boldsymbol{\gamma}} = \underset{\boldsymbol{\gamma}\in\mathbb{R}^q}{\operatorname{argmin}} \sum_{\lceil (l+1)/m \rceil}^{n} \left(\alpha(B) y_t - \boldsymbol{f}_{\boldsymbol{\gamma}}(L)^\top \boldsymbol{x}_{t,0} \right)^2, \tag{5}$$

where the lag polynomial of constrained parameters is defined by

$$\boldsymbol{f}_{\boldsymbol{\gamma}}(z) = \sum_{j=0}^{l} \boldsymbol{f}_{\boldsymbol{\gamma},j} z^j$$

with

$$\boldsymbol{f}_{\boldsymbol{\gamma},j} = \Big(f_0(\boldsymbol{\gamma}_0; j), \dots, f_i(\boldsymbol{\gamma}_i; j), \dots, f_k(\boldsymbol{\gamma}_k; j) \Big)^\top$$

for each $(i, j) \in \{0, 1, \dots, k\} \times \{0, 1, \dots, l\}$. A number of numerical algorithms are readily available in R. By default, the `optim` optimization function is used with optional choices of optimization algorithms in it. However, a user can also choose within the function `midas_r` other procedures available in R such as `nls`, customizing the desired algorithm which is suitable for the problem at hand.

The efficiency of the estimator and consistency of the standard errors depend on whether the errors of the model are spherical. We leave the aspect of efficiency of estimation to be considered by a user, however the implementation of heteroscedasticity and autocorrelation (HAC) robust standard errors is an option available in the package **sandwich** (see Zeileis 2004).

If all the functional relations $f_i(\cdot)$ were non-constraining identity mappings, then the NLS estimator would be equivalent to the ordinary least squares (OLS) problem in terms of the original parameters. For convenience, such a U-MIDAS version can be dealt with directly using a different function `midas_u` of the package (see an illustration in section 3) or a standard `lm` function, provided the alignment of data frequencies is performed as discussed in the previous section.

2.3. Taxonomy of aggregates-based MIDAS regression models

Based on the parsimony of representation argument, the higher-frequency part of conditional expectation of MIDAS regressions is often formulated in terms of aggregates as follows

$$\begin{aligned} \boldsymbol{\beta}(L)^\top \boldsymbol{x}_{t,0} &= \sum_{i=0}^{k} \sum_{j=0}^{l} \beta_j^{(i)} x_{tm_i-j}^{(i)} \\ &= \sum_{i=0}^{k} \sum_{r=0}^{q} \lambda_r^{(i)} \tilde{x}_{t-r}^{(i)}, \end{aligned} \tag{6}$$

with some low-frequency number of lags $q \in \mathbb{N}$ and parameter-driven low-frequency aggregates

$$\tilde{x}_{t-r}^{(i)} := x_{t-r}^{(i)}(\boldsymbol{\delta}_{i,r}) = \sum_{s=1}^{m_i} w_r^{(i)}(\boldsymbol{\delta}_{i,r}; s) x_{(t-1-r)m_i+s}^{(i)}$$

which depend on a weighting (aggregating within a low-frequency period) function $w_r(\boldsymbol{\delta}_{i,r}; s)$ with its parameter vector $\boldsymbol{\delta}_{i,r}$, which, in the general case, can vary with each variable/frequency and/or the low-frequency lag order $r \in \mathbb{N}$. Here the aggregation weights are usually non-negative and, for identification of parameters $\{\lambda_r^{(i)}\}_{i=0,r=0}^{h,p}$, satisfy the normalization constraint such as $\sum_{s=0}^{m_i-1} w_r(\boldsymbol{\delta}_{i,r}; s) = 1$. To have the weights add to one, it is convenient to define a weighting function in the following form

$$\forall\ i, r \quad w_r^{(i)}(\boldsymbol{\delta}_{i,r}; s) = \frac{\psi_r^{(i)}(\boldsymbol{\delta}_{i,r}; s)}{\sum_{j=1}^{m_i} \psi_r^{(i)}(\boldsymbol{\delta}_{i,r}; j)},\ s = 1, \ldots, m_i, \tag{7}$$

given some underlying function $\psi_r^{(i)}(\cdot)$. Provided that the latter function is non-negatively-valued (and the denominator is positive), the resulting weights in eq. (7) are also non-negative. Table 1 provides a list of some underlying functions producing, within the context of equation (7), the usual weighting schemes with non-negative weights (whenever the parameter space of underlying functions is appropriately bounded, which in some cases is also needed for identification of parameters). In order to avoid heavy notation, indices i and r—which are connected respectively with frequency/variable and the lag order—are dropped in the table.

Some other weighting functions which do not have a representation as in eq. (7) are also available in the package such as (non-normalized) `almonp` and the polynomial specification with step functions `polystep` (see Ghysels *et al.* (2006b) for further discussion of step functions).

Resulting (normalized) weighting scheme	$\psi(\boldsymbol{\delta}; s) := \psi_r^{(i)}(\boldsymbol{\delta}_{i,r}; s)$	**Related midasr function**
Exponential Almon lag polynomial	$\psi(\boldsymbol{\delta}; s) = \exp\left(\sum_{j=1}^{p} \delta_j s^j\right)$, $p \in \mathbb{N}$, where $\boldsymbol{\delta} = (\delta_1, \ldots, \delta_j, \ldots, \delta_p)^\top \in \mathbb{R}^p$.	`nealmon`
Beta (analogue of probability density function)	$\psi(\boldsymbol{\delta}; s) = x_s^{\delta_1 - 1}(1 - x_s)^{\delta_2 - 1}$, where $x_s := \xi + (1 - \xi)h(s)$, $h(s) := (s - 1)/(m - 1)$, with some marginally small quantity $\xi > 0$, and $\boldsymbol{\delta} = (\delta_1, \delta_2)^\top \in \mathbb{R}_+^2$.	`nbeta`
Gompertz (analogue of probability density function)	$\psi(\boldsymbol{\delta}; s) = z(s)e^{-\delta_1 z(s)}$, where $z(s) = \exp(\delta_2 s)$, and $\boldsymbol{\delta} = (\delta_1, \delta_2)^\top \in \mathbb{R}_+^2$.	`gompertzp`
Log-Cauchy (analogue of probability density function)	$\psi(\boldsymbol{\delta}; s) = s^{-1}\left(\delta_2^2 + (\ln s - \delta_1)^2\right)^{-1}$, where $\boldsymbol{\delta} = (\delta_1, \delta_2)^\top \in \mathbb{R} \times \mathbb{R}_+$.	`lcauchyp`
Nakagami (analogue of probability density function)	$\psi(\boldsymbol{\delta}; s) = s^{2\delta_1 - 1}\exp(-\delta_1/\delta_2 s^2)$, where $\boldsymbol{\delta} = (\delta_1, \delta_2)^\top$, $\delta_1 \geq 0.5, \delta_2 \in \mathbb{R}_+$.	`nakagamip`

Table 1: A sample of weighting schemes in aggregation-based MIDAS specifications.

However, the choice of a particular weighting function in the MIDAS regression with aggregates represents only one restriction imposed on $\boldsymbol{\beta}(L)$ out of many other choices to be made. To see this, let us note that aggregates-based MIDAS regressions can be connected with the following restrictions on the conditional expectation of model (3):

$$
\begin{aligned}
E\left(\alpha(B)y_t|\boldsymbol{y}_{t,1}, \{\boldsymbol{x}_{t,0}^{(i)}\}_{j=0}^{l}\right) &= \boldsymbol{\beta}(L)^\top \boldsymbol{x}_{t,0} \\
&= \sum_{i=0}^{k}\sum_{r=0}^{q} \lambda_r^{(i)} \tilde{x}_{t-r}^{(i)}, \\
&= \sum_{i=0}^{k}\sum_{r=0}^{q} \lambda_r^{(i)} \sum_{s=1}^{m_i} w_r^{(i)}(\boldsymbol{\delta}_{i,r}; s) x_{(t-1-r)m_i+s}^{(i)}, \\
\Big|_{w_r^{(i)}(\cdot)=w_r(\cdot)} &= \sum_{i=0}^{k}\sum_{r=0}^{q} \lambda_r^{(i)} \sum_{s=1}^{m_i} w_r(\boldsymbol{\delta}_{i,r}; s) x_{(t-1-r)m_i+s}^{(i)}, \\
\Big|_{w_r(\cdot)=w(\cdot)} &= \sum_{i=0}^{k}\sum_{r=0}^{q} \lambda_r^{(i)} \sum_{s=1}^{m_i} w(\boldsymbol{\delta}_{i,r}; s) x_{(t-1-r)m_i+s}^{(i)}, \\
\Big|_{\boldsymbol{\delta}_{i,r}=\boldsymbol{\delta}_i} &= \sum_{i=0}^{k}\sum_{r=0}^{q} \lambda_r^{(i)} \sum_{s=1}^{m_i} w(\boldsymbol{\delta}_i; s) x_{(t-1-r)m_i+s}^{(i)}, \\
\Big|_{\lambda_r^{(i)}=\lambda^{(i)}} &= \sum_{i=0}^{k} \lambda^{(i)} \sum_{r=0}^{q}\sum_{s=1}^{m_i} w(\boldsymbol{\delta}_i; s) x_{(t-1-r)m_i+s}^{(i)},
\end{aligned}
\tag{8}
$$

where $\boldsymbol{y}_{t,1} = (y_{t-1}, ..., y_{t-p})^\top$.

As can be seen—and leaving aside other less intuitive restrictions—depending on the choice of a particular MIDAS specification with aggregates, it can impose restrictions on the equality of

- the applied weighting scheme/function across variables and/or frequencies ($\forall\, i,\ \ w_r^{(i)}(\cdot) = w_r(\cdot)$);
- the applied weighting scheme/function across all low-frequency lags $r = 0, 1, \ldots, q$ of aggregates ($\forall\ r,\ \ w_r(\cdot) = w(\cdot)$);
- parameters of the weighting functions in each lag ($\forall\ r,\ \ \boldsymbol{\delta}_{i,r} = \boldsymbol{\delta}_i$);
- impact of contemporaneous and lagged aggregates for all lags ($\forall\ r,\ \ \lambda_r^{(i)} = \lambda^{(i)}$).

Furthermore, let s_i stand for an enumerator of i^{th} higher-frequency periods within a low-frequency period. Then, noting that, given a frequency ratio m_i, there is a one-to-one mapping between higher-frequency index $j \in \mathbb{N}$ and a pair $(r, s_i) \in \mathbb{N} \times \{1, 2, \ldots, m_i\}$

$$j = rm_i + s_i,$$

it holds

$$f_i(\boldsymbol{\gamma}_i; rm_i + s_i) = \lambda_r^{(i)} w_r^{(i)}(\boldsymbol{\delta}_{i,r}; s). \tag{9}$$

Hence, it is easy to see that the aggregates-based MIDAS induces a certain periodicity of the functional constraint f_i in eq. (3) as illustrated bellow using a stylized case where all the restrictions are imposed in eq. (8):

$$\begin{array}{llll|llll|l} f_i(\cdot,0), & f_i(\cdot,1), & \ldots & f_i(\cdot,m-1) & f_i(\cdot,m), & f_i(\cdot,m+1), & \ldots & f_i(\cdot,2m-1) & \ldots \\ \lambda^{(i)} w(\cdot,1), & \lambda^{(i)} w(\cdot,2), & \ldots & \lambda^{(i)} w(\cdot,m) & \lambda^{(i)} w(\cdot,1), & \lambda^{(i)} w(\cdot,2), & \ldots & \lambda^{(i)} w(\cdot,m) & \ldots \end{array},$$

for any $i \in \{0, 1, \ldots, h\}$. From eq. (9) it is clear that any specification of MIDAS regression models which relies on aggregates is a special case of representation (3) with just a specific functional constraint on parameters. On the other hand, not every general constraint $\boldsymbol{\beta}(L)$ can be represented using periodic aggregates. For instance, in the above characterized example the correspondence necessarily breaches whenever there exists at least one frequency i, for which none of $q \in \mathbb{N}$ satisfies $l = qm_i - 1$.

2.4. Alternative representations of MIDAS regressions

The model (1) represents a very general MIDAS regression representation. We give below a sample of other popular MIDAS regression specifications from Andreou *et al.* (2011). These specification assume that only one high frequency variable is available. We denote its frequency by m.

1. DL-MIDAS(p_X):

$$y_{t+1} = \mu + \sum_{r=0}^{p_X} \sum_{j=0}^{m-1} \beta_{rm+j} x_{(t-r)m-j} + \varepsilon_{t+1}$$

2. ADL-MIDAS(p_Y,p_X):

$$y_{t+1} = \mu + \sum_{j=0}^{p_Y} \mu_j y_{t-j} + \sum_{r=0}^{p_X} \sum_{j=0}^{m-1} \beta_{rm+j} x_{(t-r)m-j} + \varepsilon_{t+1}$$

3. FADL-MIDAS(p_F,p_X,p_Y):

$$y_{t+1} = \mu + \sum_{i=0}^{p_F} \alpha_i F_{t-i} + \sum_{j=0}^{p_Y} \mu_j y_{t-j} + \sum_{r=0}^{p_X} \sum_{j=0}^{m-1} \beta_{rm+j} x_{(t-r)m-j} + \varepsilon_{t+1},$$

where F_t is a factor derived from additional data.

4. ADL-MIDAS-M(p_X,p_Y), or multiplicative MIDAS:

$$y_{t+1} = \mu + \sum_{j=0}^{p_Y} \mu_j y_{t-j} + \sum_{r=0}^{p_X} \alpha_r X_{t-r} + \varepsilon_{t+1},$$

where $X_{t-r} = \sum_{j=0}^{m_i-1} \beta_j x_{(t-r)m-j}$.

5. MIDAS with leads, where it is assumed that we have only $J < m$ observations of high frequency variable x_τ are available for period $t+1$.

$$y_{t+1} = \mu + \sum_{j=0}^{p_Y} \mu_j Y_{t-j} + \sum_{j=1}^{J} \beta_{-j} x_{tm+j} + \sum_{r=0}^{p_X} \sum_{j=0}^{m-1} \beta_{rm+j} x_{(t-r)m-j} + \varepsilon_{t+1}$$

The parametric restriction is usually placed on coefficients β_j in these representations in the usual manner of (2), i.e. it is assumed that $\beta_j = f(\boldsymbol{\gamma}, j)$, for chosen function f with parameters $\boldsymbol{\gamma}$.

If we compare these representations with (1) the notable differences are the specification of y_{t+1} instead of y_t as a response variable and a preference to make maximum high frequency lag be a multiple of the corresponding frequency. The multiplicative MIDAS is the special case of aggregates based MIDAS specification. It is evident then that all these representations are special cases of (1).

Various examples illustrating MIDAS regressions specifications together with corresponding R code are also given in the table 3.

2.5. Specification selection and adequacy testing

Besides the usual considerations about the properties of the error term, there are two main questions about the specification of the MIDAS regression models. First, suitable functional constraints need to be selected, since their choice will affect the precision of the model. Second, the appropriate maximum lag orders need to be chosen.

One way to address both issues together is to use some information criterion to select the best model in terms of the parameter restriction and the lag orders using either in- or out-of-sample precision measures. Functions `midas_r_ic_table` and `amidas_table` of the package allow the user to make an in-sample choice using some usual information criteria, such as AIC and BIC, and a user-specified list of functional constraints.[4]

Another way is to test the adequacy of the chosen functional constraints. For instance, whenever the autoregressive terms in model (3) are present ($p > 0$), it was pointed out by Ghysels

[4] Although aimed at forecasting, the function `select_and_forecast` can also be used to perform the selection of models relying on their out-of-sample performance.

et al. (2006b) that, in the general case, $\boldsymbol{\phi}(L) = \boldsymbol{\beta}(L)/\alpha(B)$ will have seasonal pattern thus corresponding to some seasonal impact of explanatory variables on the dependent one in a pure distributed lag model (i.e., without autoregressive terms). To avoid such an effect whenever it is not (or is believed to be not) relevant, Clements and Galvão (2008) proposed to us a common factor restriction which can be formulated as a common polynomial restriction with a constraint on the polynomial $\boldsymbol{\beta}(L)$ to satisfy a factorization $\boldsymbol{\beta}(L) = \alpha(B)\boldsymbol{\phi}(L)$, so that inverting equation (3) in terms of the polynomial $\alpha(B)$ leaves $\boldsymbol{\phi}(L)$ unaffected i.e., without creating/destroying any (possibly absent) seasonal pattern of the impact of explanatory variables. However, there is little if any knowledge a priori whether the impact in the distributed lag model should be seasonal or not. Hence, an explicit testing of adequacy of the model and, in particular, of the imposed functional constraint is obviously useful.

Let $\boldsymbol{\beta}$ denote a vector of all coefficients of polynomial $\boldsymbol{\beta}(z)$ defined in eq. (3), while $\boldsymbol{f}_{\boldsymbol{\gamma}}$ stand for the corresponding vector of coefficients restricted by a (possibly incorrect) functional constraint in $\boldsymbol{f}_{\boldsymbol{\gamma}}(z)$. Let $\widehat{\boldsymbol{\beta}}$ denote the respective OLS estimates of unconstrained model i.e., where functional restrictions of parameters are *not* taken into account. Let $\hat{\boldsymbol{f}}_{\boldsymbol{\gamma}} := \boldsymbol{f}_{\boldsymbol{\gamma}}\big|_{\boldsymbol{\gamma}=\widehat{\boldsymbol{\gamma}}}$ denote a vector of the corresponding quantities obtained from the restricted model relying on the NLS estimates $\widehat{\boldsymbol{\gamma}}$ as defined in eq. (5). Denote by $\boldsymbol{\alpha}$, $\widehat{\boldsymbol{\alpha}}$, and $\widehat{\boldsymbol{\alpha}}_{\gamma}$ the corresponding vectors of coefficients of polynomial $\alpha(z)$, its OLS estimates in an unrestricted model, and its NLS estimates in a restricted model.[5] Let $\boldsymbol{\theta} := (\boldsymbol{\alpha}^\top, \boldsymbol{\beta}^\top)^\top$, $\widehat{\boldsymbol{\theta}} := (\widehat{\boldsymbol{\alpha}}^\top, \widehat{\boldsymbol{\beta}}^\top)^\top$, and $\widetilde{\boldsymbol{\theta}} := (\widehat{\boldsymbol{\alpha}}_{\boldsymbol{\gamma}}^\top, \hat{\boldsymbol{f}}_{\boldsymbol{\gamma}}^\top)^\top$ signify the corresponding vectors of all coefficients in eq. (3). Then, under the null hypothesis of $\exists\ \boldsymbol{\gamma} \in \mathbb{R}^q$ such that $\boldsymbol{f}_{\boldsymbol{\gamma}} = \boldsymbol{\beta}$, it holds

$$(\widehat{\boldsymbol{\theta}} - \widetilde{\boldsymbol{\theta}})^\top \boldsymbol{A} (\widehat{\boldsymbol{\theta}} - \widetilde{\boldsymbol{\theta}}) \sim \chi^2\big(d - q\big),$$

where $\boldsymbol{A}$ is a suitable normalization matrix (see Kvedaras and Zemlys 2012 for a standard and Kvedaras and Zemlys 2013 for a HAC-robust versions of the test), and $q = \dim(\boldsymbol{\gamma})$ and $d = \dim(\boldsymbol{\theta})$ stand for the number of parameters in a restricted and unrestricted models, respectively. Functions `hAh_test` and `hAhr_test` of the package implement the described testing as will be illustrated later.

2.6. Forecasting

Let us write model (3) for period $t+1$ as

$$y_{t+1} = \boldsymbol{\alpha}^\top \boldsymbol{y}_{t,0} + \boldsymbol{\beta}(L)^\top \boldsymbol{x}_{t+1,0} + \varepsilon_{t+1}, \tag{10}$$

where $\boldsymbol{y}_{t,0} = (y_t, \ldots, y_{t-p+1})^\top$ and $\boldsymbol{\alpha} = (\alpha_1, \alpha_2, \ldots, \alpha_p)^\top$ is a vector of parameters of the autoregressive terms. This representation is well suited for (one step ahead) conditional forecasting of y_{t+1}, provided that the information on the explanatory variables is available. If it were absent, forecasts of $\boldsymbol{x}_{t+1,0}$ would be also necessary from a joint process of $\{y_t, \boldsymbol{x}_{t,0}\}$ which might be difficult to specify and estimate correctly, especially, bearing in mind the presence of data with mixed frequencies. Instead, a direct approach to forecasting is often applied in the MIDAS framework. Namely, given an information set available up to a moment t defined by $\mathcal{I}_{t,0} = \{\boldsymbol{y}_{t,j}, \boldsymbol{x}_{t,j}\}_{j=0}^{\infty}$, where

$$\begin{aligned}
\boldsymbol{y}_{t,j} &= (y_{t-j}, ..., y_{t-j-p+1})^\top \\
\boldsymbol{x}_{t,j} &= (x_{tm_0}^{(0)}, ..., x_{tm_i}^{(i)}, ..., x_{tm_h}^{(h)})^\top,
\end{aligned}$$

[5] Recall that unconstrained $\boldsymbol{\alpha}$ elements make a subset of parameter vector $\boldsymbol{\gamma}$ of a constrained model.

an ℓ-step ahead direct forecast

$$\tilde{y}_{t+\ell} = E\left(y_{t+\ell}|\mathcal{I}_{t,0}\right) = \boldsymbol{\alpha}_\ell^\top \boldsymbol{y}_{t,0} + \boldsymbol{\beta}_\ell(L)^\top \boldsymbol{x}_{t,0},\ \ell \in \mathbb{N}, \tag{11}$$

can be formed leaning on a model linked to a corresponding conditional expectation

$$y_{t+\ell} = \boldsymbol{\alpha}_\ell^\top \boldsymbol{y}_{t,0} + \boldsymbol{\beta}_\ell(L)^\top \boldsymbol{x}_{t,0} + \varepsilon_{\ell,t},\ E\left(\varepsilon_{\ell,t}|\mathcal{I}_{t,0}\right),$$

where $\boldsymbol{\alpha}_\ell$ and $\boldsymbol{\beta}_\ell(L)$ are the respective horizon ℓ-specific parameters. Note that, in principle, these conditional expectations have a form of representation (3) with certain restrictions on the original lag polynomials of coefficients. Hence, in the general case, the suitable restrictions for each ℓ will have a different form.

Given periods $\ell = 1, 2, \ldots$, and a selected model or a list of specifications to be considered, package **midasr** provides the point forecasts corresponding to the estimated analogue of eq. (11) evaluates the precision of different specifications, and performs weighted forecasting using the framework defined in Ghysels (2013).

3. Implementation in midasr package

3.1. Data handling

From a data handling point of view, the key specificity of the MIDAS regression model is that the length of observations of variables observed at various frequencies differs and needs to be aligned as described in Section 2. There is no existing R function which performs such a transformation and the package **midasr** provides a solution to these challenges. The basic functionality of data handling is summarized in Table 2.

Function	Description	Example	Notes
`mls(x, k, m)`	Stacks a HF data vector x into a corresponding matrix of observations at LF of size $\frac{\dim x}{m} \times \dim k$: from the first to the last HF lag defined by vector k.	`mls(x, 2:3, 3)`	$\frac{\dim x}{m}$ must be an integer (NA are allowed). For $m = 1$, the function produces lags of x that are defined by vector argument k, e.g., `mls(x, 2:3, 1)` yields a dataset containing the lags x_{t-2} and x_{t-3} of x_t.
`fmls(x, k, m)`	Same as `mls`, except that k is a scalar and the $k+1$ lags are produced starting from 0 up to k.	`fmls(x, 2, 3)`	`fmls(x, 2, 3)` is equivalent to `mls(x, 0:2, 3)`.
`dmls(x, k, m)`	Same as `fmls`, only the resulting matrix contains $k+1$ first-order HF differences of x.	`dmls(x, 2, 3)`	`mls(x, 1, 1)` can be used in `dmls` to get stacking of lagged differences, e.g., `dmls(mls(x, 1, 1), 2, 3)`.

Table 2: A summary of a basic data handling functionality in the package **midasr**.

Function `fmls(x,k,m)` performs exactly the transformation defined in equation (4), converting an observation vector x of a given (potentially) higher-frequency series into the corresponding stacked matrix of observations of $(k+1)$ low-frequency series (contemporaneous with k lags) as defined by the maximum lag order k and the frequency ratio m. For instance, given a series of twelve observations

```
R> x <- 1:12
```

we get the following result

```
R> fmls(x, k = 2, m = 3)
```

```
     X.0/m X.1/m X.2/m
[1,]     3     2     1
[2,]     6     5     4
[3,]     9     8     7
[4,]    12    11    10
```

i.e., three variables (a contemporaneous and two lags) with four low-frequency observations $(n = 12/m)$.

Function `mls` is slightly more flexible as the lags included can start from a given order rather than from zero, whereas the function `fmls` uses a full lag structure. `dmls` performs in addition a first-order differencing of the data which is convenient when working with integrated series.

A couple of issues should be taken into account when working with series of different frequencies.

- It is assumed that the numbers of observations of different frequencies match exactly through the frequency ratio $(n_i = nm_i)$, and the first and last observations of each series of different frequency are correspondingly aligned (possibly using `NA` to account for some missing observations for series of higher frequency).
- Because of different lengths of series of various frequencies, the data for the model cannot be kept in one `data.frame`. It is expected that variables for the model are either vectors residing in R global environment, or are passed as elements of a `list`. Variables of different frequency can be in the same `data.frame`, which in turn should be an element of a list.

3.2. An example of simulated MIDAS regression

Using the above data handling functions, it is straightforward to simulate a response series from the MIDAS regression as a data generating process (DGP). For instance, suppose one is willing to generate a low-frequency response variable y in the MIDAS with two higher-frequency series x and z where the impact parameters satisfy the exponential Almon lag polynomials of different orders as follows:

$$\begin{aligned} y_t &= 2 + 0.1t + \sum_{j=0}^{7} \beta_j^{(1)} x_{4t-j} + \sum_{j=0}^{16} \beta_j^{(2)} z_{12t-j} + \varepsilon_t, \\ x_{\tau_1} &\sim n.i.d.(0,1), \quad z_{\tau_2} \sim n.i.d.(0,1), \quad \varepsilon_t \sim n.i.d.(0,1), \end{aligned} \tag{12}$$

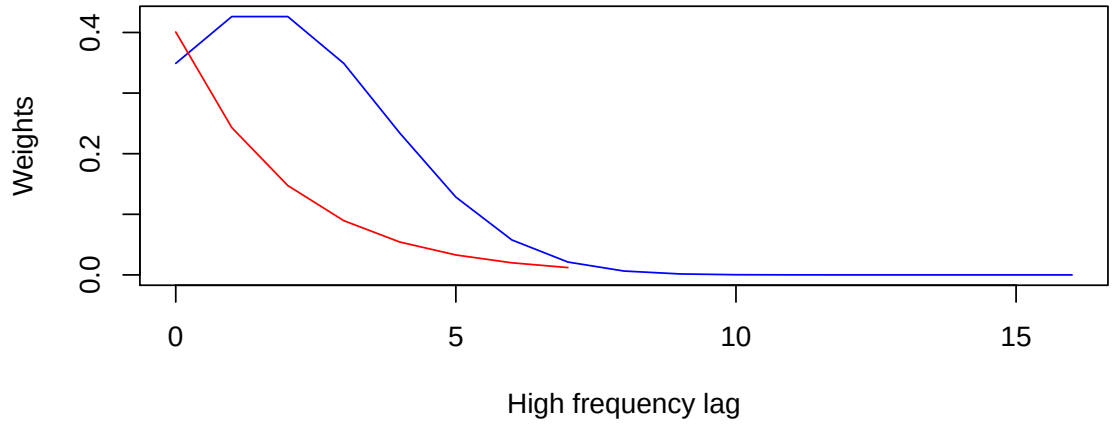

Figure 2: A plot of shapes of functional constraints of x_τ (blue) and z_τ (red).

where $(x_{\tau_1}, z_{\tau_2}, \varepsilon_t)$ are independent for any $(\tau_1, \tau_2, t) \in \mathbb{Z}^3$, and

$$\beta_j^{(i)} = \gamma_0^{(i)} \frac{\exp\left(\sum_{s=1}^{q_i-1} \gamma_s^{(i)} j^s\right)}{\sum_{j=0}^{d_i-1} \exp\left(\sum_{s=1}^{q_i-1} \gamma_s^{(i)} j^s\right)}, \quad i = 1, 2,$$

where $d_1 = k_1 + 1 = 8$ is a multiple of the frequency ratio $m_1 = 4$, whereas $d_2 = k_2 + 1 = 17$ is not a multiple of $m_2 = 12$. Here $q_1 = 2$, $q_2 = 3$ with parameterizations

$$\boldsymbol{\gamma}_1 = (1, -0.5)^\top,$$
$$\boldsymbol{\gamma}_2 = (2, 0.5, -0.1)^\top,$$

which yield the shapes of functional constraints as plotted in Figure 2.

The following R code produces a series according to the DGP characterized above:

```
R> set.seed(1001)
R> n <- 250
R> trend <- c(1:n)
R> x <- rnorm(4 * n)
R> z <- rnorm(12 * n)
R> fn_x <- nealmon(p = c(1, -0.5), d = 8)
R> fn_z <- nealmon(p = c(2, 0.5, -0.1), d = 17)
R> y <- 2 + 0.1 * trend + mls(x, 0:7, 4) %*% fn_x + mls(z, 0:16,
+     12) %*% fn_z + rnorm(n)
```

It is of interest to note that the impact of variable x can be represented using aggregates-based MIDAS, whereas the impact of z cannot.

3.3. Some specification examples of MIDAS regressions

Suppose now that we have (only) observations of y, x, and z which are stored as vectors, matrices, time series, or list objects in R, and our intention is to estimate a MIDAS regression model as in equation (12):

a) without restricting the parameters (as in U-MIDAS) and using the OLS;

b) with the exponential Almon lag polynomial constraint on parameters (as in the function `nealmon`) and using the NLS.

The OLS estimation as in case a) is straightforwardly performed using

```
R> eq.u <- lm(y ~ trend + mls(x, k = 0:7, m = 4) + mls(z, k = 0:16,
+     m = 12))
```

or, equivalently

```
R> eq.u <- midas_r(y ~ trend + mls(x, 0:7, 4) + mls(z, 0:16, 12),
+     start = NULL)
```

Note that in this case, `midas_r` picks up the variables from the global R environment. It is possible to pass the data explicitly:

```
R> eq.u <- midas_r(y ~ trend + mls(x, 0:7, 4) + mls(z, 0:16, 12),
+     start = NULL, data = list(y = y, trend = trend, x = x, z = z))
```

The variables of the same frequency can reside in the same `data.frame`:

```
R> eq.u <- midas_r(y ~ trend + mls(x, 0:7, 4) + mls(z, 0:16, 12),
+     start = NULL, data = list(data.frame(y = y, trend = trend),
+         x = x, z = z))
```

In this case, there is no need to name the `data.frame` element in the list.

The following R code estimates the constrained case b) using the function `midas_r` and reports the NLS estimates $\widehat{\gamma}$ of parameters with the related summary statistics.

```
R> eq.r <- midas_r(y ~ trend + mls(x, 0:7, 4, nealmon) + mls(z,
+     0:16, 12, nealmon), start = list(x = c(1, -0.5), z = c(2,
+     0.5, -0.1)))
R> summary(eq.r)


 Formula y ~ trend + mls(x, 0:7, 4, nealmon) + mls(z, 0:16, 12, nealmon)

 Parameters:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  1.988196   0.115299   17.24  < 2e-16 ***
```

```
trend        0.099883   0.000777  128.57  < 2e-16 ***
x1           1.353343   0.151220    8.95  < 2e-16 ***
x2          -0.507566   0.096670   -5.25  3.3e-07 ***
z1           2.263473   0.172815   13.10  < 2e-16 ***
z2           0.409653   0.155685    2.63  0.00905 **
z3          -0.072979   0.020392   -3.58  0.00042 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

 Residual standard error: 0.932 on 242 degrees of freedom
```

As you can see the syntax of the function `midas_r` is similar to the standard R function `nls`. The model is specified via familiar `formula` interface. The lags included and functional restriction used can be individual to each variable and are specified within the respective `mls`, `fmls`, or `dmls` function used with `midas_r`. It is necessary to provide a list of starting values for each variable with restricted coefficients, since it implicitly defines the number of parameters of the constraint functions to be used for each series.

The main difference with the function `nls` is that there is a greater choice of numerical optimization algorithms. The function `midas_r` is written in a way that in theory it can use any R optimization function. The choice is controlled via `Ofunction` argument. Currently it is possible to use functions `optim` and `nls` which are present in a standard R installation and function `optimx` from the package **optimx** Nash and Varadhan (2011). The additional arguments to the aforementioned functions can be specified directly in the call to `midas_r`. So for example if we want to use the optimization algorithm of Nelder and Mead, which is the default option in the function `optim` we use the following code

```
R> eq_r2 <- midas_r(y ~ trend + mls(x, 0:7, 4, nealmon) +
R>                  mls(z, 0:16, 12, nealmon),
R>                  start = list(x = c(1, -0.5), z = c(2, 0.5, -0.1)),
R>                  Ofunction = "optim", method = "Nelder-Mead")
```

If we want to use Golub-Pereyra algorithm for partially linear least-squares models implemented in the function `nls` we use the following code

```
R> eq_r2 <- midas_r(y ~ trend + mls(x, 0:7, 4, nealmon) +
R>                  mls(z, 0:16, 12, nealmon),
R>                  start = list(x = c(1,-0.5), z = c(2, 0.5, -0.1)),
R>                  Ofunction = "nls", method = "plinear")
```

It is possible to re-estimate the NLS problem with the different algorithm using as starting values the final solution of the previous algorithm. For example it is known, that the default algorithm in `nls` is sensitive to starting values. So first we can use the standard Nelder-Mead algorithm to find "more feasible" starting values and then use the `nls` to get the final result:

```
R> eq_r2 <- midas_r(y ~ trend + mls(x, 0:7, 4, nealmon) +
R>                  mls(z, 0:16, 12, nealmon),
R>                  start = list(x = c(1, -0.5), z = c(2, 0.5, -0.1)),
R>                  Ofunction = "optim", method = "Nelder-Mead")
R> eq_r2 <- update(eq_r2, Ofunction = "nls")
```

The output of the optimization function used can be found by inspecting the element `opt` of `midas_r` output.

```
R> eq_r2 <- midas_r(y ~ trend + mls(x, 0:7, 4, nealmon) +
+                   mls(z, 0:16, 12, nealmon),
+                   start = list(x = c(1, -0.5), z = c(2, 0.5, -0.1)),
+                   Ofunction = "optim", method = "Nelder-Mead")
R> eq_r2$opt

$par
(Intercept)       trend          x1          x2          z1          z2
    1.98565     0.09990     1.35252    -0.50816     2.26318     0.40886
         z3
   -0.07284

$value
[1] 210

$counts
function gradient
     502       NA

$convergence
[1] 1

$message
NULL
```

Here we observe that the Nelder-Mead algorithm evaluated the cost function 502 times.

The optimization functions in R report the status of the convergence of optimization algorithm by the numeric constant, 0 indicating succesful convergence. This code is reported as the element `convergence` of the `midas_r` output.

```
R> eq_r2$convergence

[1] 1
```

In this case the convergence was not successfull. The help page of the function `optim` indicates that convergence code 1 means that the iteration limit was reached.

In order to improve the convergence it is possible to use user defined gradient functions. To use them it is necessary to define gradient function of the restriction. For example for the `nealmon` restriction the gradient function is defined in the following way:

```
R> nealmon_gradient <- function(p, d, m) {
+     i <- 1:d
+     pl <- poly(i, degree = length(p) - 1, raw = TRUE)
```

```
+     eplc <- exp(pl %*% p[-1])[, , drop=TRUE]
+     ds <- apply(pl * eplc, 2, sum)
+     s <- sum(eplc)
+     cbind(eplc/s, p[1] * (pl * eplc/s - eplc %*% t(ds)/s^2))
+ }
```

The gradient functions are passed as named list elements via argument `weight_gradients` :

```
R> eq_r2 <- midas_r(y ~ trend + mls(x, 0:7, 4, nealmon) +
+                   mls(z, 0:16, 12, nealmon),
+                   start = list(x = c(1, -0.5), z = c(2, 0.5, -0.1)),
+                   weight_gradients=list(nealmon = nealmon_gradient))
```

This way `midas_r` calculates the exact gradient of the NLS problem (5) using the specified gradient function of the restriction. For all the types of the restrictions referenced in table 3 their gradient functions are specified in the package **midasr**. The naming convention for gradient functions is `restriction_name_gradient`. It is not necessary to explicitly pass gradient functions named according to this convention. If `weight_gradients` is not `NULL` and does not contain the apropriately named element, it is assumed that there exists a gradient function conforming to the gradient naming convention which is then subsequently used.

The gradient and the hessian of the NLS problem are supplied as the output of `midas_r`. The gradient is calculated exactly if appropriate gradients for weight functions are supplied as explained above, otherwise the numerical approximation of the gradient is calculated using the package **numDeriv** Gilbert and Varadhan (2012). For hessian the numerical approximation is always used. Having the gradient and hessian calculated allows to check whether the necessary and sufficient conditions for the convergence are satisfied. This is performed by the function `deriv_test` which calculates the Euclidean norm of the gradient and the eigenvalues of the hessian. It then tests whether the norm of gradient is close to zero and whether the eigenvalues are positive.

```
R> deriv_tests(eq.r, tol = 1e-06)

$first
[1] FALSE

$second
[1] TRUE

$gradient
[1]  0.0042441  0.0503346 -0.0008513  0.0021252  0.0004805 -0.0004506
[7] -0.3851794

$eigenval
[1] 1.048e+07 5.888e+04 3.664e+02 1.221e+02 8.117e+01 5.148e+01 4.596e+01
```

To retrieve a vector of constrained estimates $\tilde{\theta}$ (and, hence, also $\hat{\boldsymbol{f}} = \boldsymbol{f}_{\boldsymbol{\gamma}}\big|_{\boldsymbol{\gamma}=\hat{\boldsymbol{\gamma}}}$) which.min corresponds to the vector $\boldsymbol{\theta}$ ($\boldsymbol{\beta}$, respectively), the function `coef` can be used as follows

```
R> coef(eq.r, midas = TRUE)

(Intercept)       trend          x1          x2          x3          x4
  1.988e+00   9.988e-02   5.481e-01   3.300e-01   1.986e-01   1.196e-01
         x5          x6          x7          x8          z1          z2
  7.197e-02   4.332e-02   2.608e-02   1.570e-02   3.347e-01   4.050e-01
         z3          z4          z5          z6          z7          z8
  4.235e-01   3.827e-01   2.989e-01   2.018e-01   1.177e-01   5.932e-02
         z9         z10         z11         z12         z13         z14
  2.584e-02   9.728e-03   3.165e-03   8.898e-04   2.162e-04   4.539e-05
        z15         z16         z17
  8.237e-06   1.292e-06   1.750e-07
```

In the example provided above, a functional constraint was imposed directly on $\boldsymbol{\beta}(L)$ terms corresponding to each series without the usage of aggregates. Relying on the relationship (9), it is always possible to write such an explicit general constraint from an aggregates-based one. For convenience of a user, the function `amweights` can be used to form several standard periodic functional constraints with 'typical' restrictions explicated in equation (7). For instance,

```
R> amweights(p = c(1, -0.5), d = 8, m = 4, weight = nealmon, type = "C")

[1] 0.4551 0.2760 0.1674 0.1015 0.4551 0.2760 0.1674 0.1015
```

with `type="C"` corresponds to a fully restricted version of aggregates-based expression (7) apart the cross-restriction on the equality of weighting schemes between different variables/frequencies. Note that the code above repeats the result of

```
R> nealmon(p = c(1, -0.5), d = 4)

[1] 0.4551 0.2760 0.1674 0.1015
```

twice ($d/m = 2$), as implied by the number of periods at higher-frequency (`d=8`) and the frequency ratio (`m=4`). In this way, the function `amweights` can be used to define explicitly a new functional constraint relying on the relationship (9). Alternatively, one can indicate directly within the function `midas_r` that the aggregates-based restriction must be used as follows

```
R> eq_r2 <- midas_r(y ~ trend + mls(x, 0:7, 4, amweights, nealmon,
+     "C") + mls(z, 0:16, 12, nealmon), start = list(x = c(1, -0.5),
+     z = c(2, 0.5, -0.1)))
```

where the first variable follows and aggregates-based MIDAS restriction scheme. Note that the selection of alternative types "A" and "B" are connected with specifications having a larger number of parameters (see Table 3), hence the list of starting values needs to be adjusted to account for an increase in the number of (potentially unequal) impact parameters.

It should be also noted that, whenever the aggregates-connected restrictions are used, the number of periods must be a multiple of the frequency ratio. For instance, the current lag specification for variable z is not consistent with this requirement and cannot be represented through the (periodic) aggregates, but either `mls(z,0:11,12,amweights,nealmon,"C")` or `mls(z,0:23,12,amweights,nealmon,"C")` would be valid expressions from the code implementation point of view.

Table 3 summarizes and provides various other examples of correspondence between `midas_r` coding and the analytical specifications of MIDAS regressions.

3.4. Adequacy testing of restrictions

Given a MIDAS regression model estimated with `midas_r`, the empirical adequacy of the functional restrictions can be tested under quite standard assumptions (see Kvedaras and Zemlys 2012 and Kvedaras and Zemlys 2013) using functions `hAh_test` and `hAhr_test` of the package. In the case of a stationary series $\{y_t\}$ they can be applied directly, whereas whenever $\{y_t\}$ is cointegrated with explanatory variables, a special transformation needs to be applied before the testing (see e.g., Bilinskas and Zemlys 2013). The `hAh_test` can be used whenever errors of the process are independently and identically distributed, whereas the `hAhr_test` uses a HAC-robust version of the test. We should just point out that, whenever no significant HAC in the residuals are observed, we would suggest using `hAh_test` which would then have more precise test sizes in small samples. In the case of integrated series $\{y_t\}$ which is cointegrated with explanatory variables, some other alternatives are available (see Bilinskas, Kvedaras, and Zemlys 2013).

For illustration, let us use the name `eq.r` of an estimated model as in the previous subsections. Then the functions produce, respectively,

```
R> hAh_test(eq.r)


        hAh restriction test

data:
hAh = 16.55, df = 20, p-value = 0.6818

R> hAhr_test(eq.r)


        hAh restriction test (robust version)

data:
hAhr = 14.85, df = 20, p-value = 0.7847
```

Here the value of a test statistic, the degrees of freedom (the number of binding constraints on parameters in equation (3)), and the empirical significance of the null hypothesis that a functional constraint is adequate are reported.

As can be seen, such a specification, which in fact corresponds to the underlying DGP, cannot be rejected at the usual significance levels, whereas e.g., reducing the number of parameters

Description	Code example	Analytical expression	Notes
Different constraint functions	`midas_r(y ~ mls(x, 0:7, 4, nealmon) + mls(z, 0:16, 12, gompertzp), start = list(x = c(1, -0.5), z = c(1, 0.5, 0.1)))`	$y_t = c + \sum_{j=0}^{7} \beta_j^{(1)} x_{4t-j} + \sum_{j=0}^{16} \beta_j^{(2)} z_{12t-j} + \varepsilon_t$	Constraints on $\beta_j^{(i)}$, $i = 1, 2$ are given by different functions.
Partial constraint (only on z)	`midas_r(y ~ mls(x, 0:7, 4) + mls(z, 0:16, 12, nealmon), start = list(z = c(1, -0.5)))`	$y_t = c + \sum_{j=0}^{7} \beta_j^{(1)} x_{4t-j} + \sum_{j=0}^{16} \beta_j^{(2)} z_{12t-j} + \varepsilon_t$	x enters linearly with unconstrained $\beta_j^{(1)}$.
With unrestricted autoregressive terms	`midas_r(y ~ mls(y, 1:2, 1) + mls(x, 0:7, 4, nealmon), start = list(x = c(1, -0.5)))`	$y_t = c + \sum_{j=1}^{2} \alpha_j y_{t-j} + \sum_{j=0}^{7} \beta_j x_{4t-j} + \varepsilon_t$	Autoregressive terms enter linearly with unconstrained coefficients.
With a common factor restriction	`midas_r(y ~ mls(y, 1:2, 1, "*") + mls(x, 0:7, 4, nealmon), start = list(x = c(1, -0.5)))`	$\alpha(B) y_t = c + \alpha(B)\lambda(L) x_{4t} + \varepsilon_t,$	Here coefficients of $\lambda(z)$ are assumed to satisfy nealmon restriction.
With autoregressive parameters restricted by a function	`midas_r(y ~ mls(y, 1:6, 1, nealmon) + mls(x, 0:7, 4, nealmon), start = list(y = c(1, -0.5), x = c(1, -0.5)))`	$y_t = c + \sum_{j=1}^{6} \alpha_j y_{t-j} + \sum_{j=0}^{7} \beta_j x_{4t-j} + \varepsilon_t$	Autoregressive parameters α_j, $j = 1, \ldots, 6$ are constrained to satisfy nealmon restriction.
Aggregates-based (Case A)	`midas_r(y ~ mls(x, 0:7, 4, amweights, nealmon, "A"), start = list(x = c(1, 1, 1, -0.5)))`	$y_t = c + \sum_{r=0}^{1} \lambda_r \sum_{s=1}^{4} w(\boldsymbol{\delta}_r; s) x_{4(t-1-r)+s} + \varepsilon_t$	The same weighting scheme (not parameters) is used in aggregation.
Aggregates-based (Case B)	`midas_r(y ~ mls(x, 0:7, 4, amweights, nealmon, "B"), start = list(x = c(1, 1, -0.5)))`	$y_t = c + \sum_{r=0}^{1} \lambda_r \sum_{s=1}^{4} w(\boldsymbol{\delta}; s) x_{4(t-1-r)+s} + \varepsilon_t$	The same weights are used in aggregation.
Aggregates-based (Case C)	`midas_r(y ~ mls(x, 0:7, 4, amweights, nealmon, "C"), start = list(x = c(1, -0.5)))`	$y_t = c + \lambda \sum_{r=0}^{1} \sum_{s=1}^{4} w(\boldsymbol{\delta}; s) x_{4(t-1-r)+s} + \varepsilon_t$	A common impact parameter of lags and the same weights are used in aggregation.
With a user-defined constraint	`midas_r(y ~ mls(x, 0:101, 4, fn), start = list(x = c(0, 0)))`	$y_t = c + \sum_{j=0}^{101} \beta_j x_{4t-j} + \varepsilon_t,$ $\beta_j = \gamma_1 (j+1)^{\gamma_2},\ j = 0, 1, \ldots, 101.$	User defined function: `fn <- function(p, d) p[1] * c(1:d)^p[2]`.

Table 3: A non-extensive list of possible specifications of the MIDAS regression in the **midasr** package.

of functional constraint of variable z to only two instead of three is quite strongly rejected using either version of the test:

```
R> eq_rb <- midas_r(y ~ trend + mls(x, 0:7, 4, nealmon) +
+                   mls(z, 0:12, 12, nealmon),
+                   start = list(x = c(1, -0.5), z = c(2, -0.1)))
R> hAh_test(eq_rb)


        hAh restriction test

data:
hAh = 36.89, df = 17, p-value = 0.00348

R> hAhr_test(eq_rb)


        hAh restriction test (robust version)

data:
hAhr = 32.88, df = 17, p-value = 0.01168
```

Whenever the empirical adequacy cannot be reject at some appropriate level of significance for a couple of models, we could further rely on information criteria to make the selection of the best candidate(s).

3.5. Model selection

Suppose that we want to investigate which out of several functional constraints—for instance, the normalized ("nealmon") or non-normalized ("almonp") exponential Almon lag polynomials, or with polynomial of order 2 or 3, and so on—are better suited in a MIDAS regression model of y on x and z (possibly different for each variable). Since the best maximum number of lags can differ with a functional constraint and/or variable/frequency, let us first define using the **midasr** function `expand_weights_lags` the sets of potential models corresponding to each explanatory variable as follows

```
R> set_x <- expand_weights_lags(weights = c("nealmon", "almonp"),
+                               from = 0, to = c(5, 10), m = 1,
+                               start = list(nealmon = c(1, -1),
+                                            almonp = c(1, 0, 0)))
R> set_z <- expand_weights_lags(c("nealmon", "nealmon"),
+                               0, c(10, 20), 1,
+                               start = list(nealmon = c(1, -1),
+                                            nealmon = c(1, -1, 0)))
```

Here, for each variable, vector (or list) `weights` defines the potential restrictions to be considered and a list `start` gives the appropriate starting values defining implicitly the number of parameters per a function.

The potential lag structures are given by the following ranges of high-frequency lags: from [from; m∗ min(to)] to [from; m∗ max(to)]. When aggregates-based modeling is involved using amweights in midas_r, m can be set to the frequency ratio which ensures that the considered models (lag structures) are multiples of it. Otherwise, we would recommend to operate with high-frequency lag structures without changing the default value $m = 1$. Then, the set of potential models is defined as all possible different combinations of functions and lag structures with a corresponding set of starting values. A simple example bellow illustrates the result in order to reveal the underlying structure, which, besides the understanding of it, is otherwise not needed for a user.

```
R> expand_weights_lags(weights = c("nealmon", "nbeta"),
+                      from = 1, to = c(2, 3), m = 1,
+                      start = list(nealmon = c(1, -1),
+                                   nbeta = rep(0.5, 3)))

  weights lags           starts
1 nealmon  1:2         c(1, -1)
2 nealmon  1:3         c(1, -1)
3   nbeta  1:2 c(0.5, 0.5, 0.5)
4   nbeta  1:3 c(0.5, 0.5, 0.5)
```

Given the sets of potential specifications for each variable as defined above, the estimation of all the models is performed by

```
R> eqs_ic <- midas_r_ic_table(y ~ trend + mls(x, 0, m = 4) + fmls(z,
+     0, m = 12), table = list(z = set_z, x = set_x))
```

The function midas_r_ic_table returns a summary table of all models together with the corresponding values of the usual information criteria and the empirical sizes of adequacy testing of functional restrictions of parameters. The result of derivative tests and the convergence status of the optimization function is also returned.

The summary table is a data.frame where each row corresponds to candidate model, so this table can be manipulated in the usual R way. The table can be accessed as table element of the list returned by midas_r_ic_table. The list of fitted midas_r objects of all candidate models can be accessed as candlist element. It is possible to inspect each candidate model and fine-tune its convergence if necessary.

```
R> eqs_ic$candlist[[5]] <- update(eqs_ic$candlist[[5]], Ofunction = "nls")
```

The summary table can be recalculated by using the update method for midas_r_ic_table. This function then recalculates all the necessary statistics.

```
R> eqs_ic <- update(eqs_ic)
```

It should be pointed out that there is no need to provide the weighting function nor a specific lag order in the mls functions in a call to midas_r_ic_table, since they are defined by the

respective potential sets of models under option `table`. Any provided values with `mls` (or other similar functions) are over-written by those defined in `table`.

Finally, the best model in terms of a selected information criterion in a restricted or unrestricted model then is simply obtained by using

```
R> modsel(eqs_ic, IC = "AIC", type = "restricted")
```

which also prints the usual summary statistics as well as the testing of adequacy of the applied functional restriction using, by default, the `hAh_test`. A word of caution is needed here to remind that, as is typical, the empirical size of a test corresponding to a complex model-selection procedure might not correspond directly to a nominal one of a single-step estimation.

3.6. Forecasting

Conditional forecasting (with confidence intervals, etc.) using unrestricted U-MIDAS regression models which are estimated using `lm` can be performed using standard R functions e.g., `predict.lm`. Conditional point prediction given a specific model is also possible relying on a standard `predict` function.

The function `predict` works similarly to `predict.lm`. It takes the new data, transforms it into an appropriate matrix and multiplies it with the coefficients. Suppose we want to produce the forecast $\hat{y}_{T+1|T}$ for the model (12). To produce this forecast we need the data $x_{4(T+1)}, ..., x_{4T-3}$ and $z_{12(T+1)}, ..., z_{12T-4}$. It would be tedious to calculate precisely the required data each time we want to perform a forecasting exercise. To alleviate this problem the package **midasr** provides the function `forecast`. This function assumes that the model was estimated with the data up to low frequency index T. It is then assumed that the new data is the data after the low frequency T and then calculates the appropriate forecast. For example suppose that we have new data for one low frequency period for the model (12). Here is how the forecast for one period would look like:

```
R> newx <- rnorm(4)
R> newz <- rnorm(12)
R> forecast(eq_rb, newdata = list(x = newx, z = newz, trend = 251))

  Point Forecast
1          28.29
```

It is also common to estimate models which do not require new data for forecasting

$$y_{t+\ell} = 2 + 0.1t + \sum_{j=0}^{7} \beta_j^{(1)} x_{4t-j} + \sum_{j=0}^{16} \beta_j^{(2)} z_{12t-j} + \varepsilon_{t+\ell},$$

where ℓ is the desired forecasting horizon. This model can be rewritten as

$$y_t = 2 + 0.1(t-\ell) + \sum_{j=4\ell}^{7+4\ell} \beta_j^{(1)} x_{4t-j} + \sum_{j=12\ell}^{16+12\ell} \beta_j^{(2)} z_{12t-j} + \varepsilon_t,$$

and can be estimated using `midas_r`. For such a model we can get forecasts $\hat{y}_{T+\ell|T}, ..., \hat{y}_{T+1|T}$ using the explanatory variable data up to low frequency index T. To obtain these forecasts using the function `forecast` we need to supply `NA` values for explanatory variables. An example for $\ell = 1$ is as follows:

```
R> eq_f <- midas_r(y ~ trend + mls(x, 4 + 0:7, 4, nealmon) + mls(z,
+     12 + 0:16, 12, nealmon), start = list(x = c(1, -0.5), z = c(2,
+     0.5, -0.1)))
R> forecast(eq_f, newdata = list(x = rep(NA, 4), z = rep(NA, 12),
+     trend = 251))

  Point Forecast
1           27.2
```

Note that we still need to specify a value for the trend.

In addition, the package **midasr** provides a general flexible environment for out-of-sample prediction, forecast combination, and evaluation of restricted MIDAS regression models using the function `select_and_forecast`. If exact models were known for different forecasting horizons, it can also be used just to report various in- and out-of-sample prediction characteristics of the models. In the general case, it also performs an automatic selection of the best models for each forecasting horizon from a set of potential specifications defined by all combinations of functional restrictions and lag orders to be considered, and produces forecast combinations according to a specified forecast weighting scheme.

In general, the definition of potential models in the function `select_and_forecast` is similar to that one uses in the model selection analysis described in the previous section. However, different best performing specifications are most likely related with each low-frequency forecasting horizon $\ell = 0, 1, 2, \ldots$. Therefore the set of potential models (parameter restriction functions and lag orders) to be considered for each horizon needs to be defined.

Suppose that, as in the previous examples, we have variables x and z with frequency ratios $m_1 = 4$ and $m_2 = 12$, respectively. Suppose that we intend to consider forecasting of y up to three low-frequency periods $\ell \in \{1, 2, 3\}$ ahead. It should be noted that, in terms of high-frequency periods, they correspond to $\ell m_1 \in \{4, 8, 12\}$ for variable x, and $\ell m_2 \in \{12, 24, 36\}$ for variable z. Thus these variable-specific vectors define the lowest lags[6] of high-frequency period to be considered for each variable in the respective forecasting model (option `from` in the function `select_and_forecast`). Suppose further that in all the models we want to consider specifications having not less than 10 high-frequency lags and not more than 15 for each variable. This defines the maximum high-frequency lag of all potential models considered for each low-frequency horizon period $\ell \in \{1, 2, 3\}$. Hence, for each variable, three corresponding pairs $(\ell m_1 + 10, \ell m_1 + 15)$, $\ell \in \{1, 2, 3\}$ will define the upper bounds of ranges to be considered (option `to` in the function `select_and_forecast`). For instance, for variable x, three pairs $(14, 19), (18, 23)$, and $(22, 27)$ correspond to $\ell = 1, 2$, and 3 and together with that defined in option `from` (see `x=(4,8,12)`) imply that the following ranges of potential models will be under consideration for variable x:

$\ell = 1:$ from $[4 - 14]$ to $[4 - 19]$,

[6] Including lags smaller than that would imply that more information on explanatory variables is available and, in fact, $\ell - 1$ forecasting horizon is actually under consideration.

$\ell = 2$: from $[8 - 18]$ to $[8 - 23]$,

$\ell = 3$: from $[12 - 22]$ to $[12 - 27]$.

The other options of the function `select_and_forecast` are options of functions `midas_r_ic_table`, `modsel` and `average_forecast`.

```
R> cbfc <- select_and_forecast(
R>             y ~ trend + mls(x, 0, 4) + mls(z, 0, 12),
R>             from = list(x = c(4, 8, 12), z = c(12, 24, 36)),
R>             to = list(x = rbind(c(14, 19), c(18, 23), c(22, 27)),
R>                       z = rbind(c(22, 27), c(34, 39), c(46, 51))),
R>             insample = 1:200, outsample = 201:250,
R>             weights = list(x = c("nealmon", "almonp"),
R>                            z = c("nealmon", "almonp")),
R>             wstart = list(nealmon = rep(1, 3), almonp = rep(1, 3)),
R>             IC = "AIC", seltype = "restricted", ftype = "fixed",
R>             measures = c("MSE", "MAPE", "MASE"),
R>             fweights = c("EW", "BICW", "MSFE", "DMSFE"))
```

The names of weighting schemes are taken from MIDAS Matlab toolbox Ghysels (2013). Similarly forecasting using rolling and recursive model estimation samples defined therein Ghysels (2013) is supported by setting option `seltype = "rolling"` or `seltype = "recursive"`.

Then, among others,

```
R> cbfc$accuracy$individual
R> cbfc$accuracy$average
```

report, respectively:

- the best forecasting equations (in terms of a specified criterion out of the above-defined potential specifications), and their in- and out-of-sample forecasting precision measures for each forecasting horizon;
- the out-of-sample precision of forecast combinations for each forecasting horizon.

The above example illustrated a general usage of the function `select_and_forecast` including selection of best models. Now suppose that a user is only interested in evaluating a one step ahead forecasting performance of a given model. Suppose further that he/she a priori knows that the best specifications to be used for this forecasting horizon $\ell = 1$ is with

- `mls(x,4:12,4,nealmon)` with parameters `x=c(2,10,1,-0.1)` (the first one representing an impact parameter and the last three being the parameters of the normalized weighting function), and
- `mls(z,12:20,12,nealmon)` with parameters `z=c(-1,2,-0.1)` i.e., with one parameter less in the weighting function.

Given already preselected and evaluated models, user can use the function `average_forecast` to evaluate the forecasting performance. To use this function at first it is necessary to fit the model and then pass it to function `average_forecast` specifying the in-sample and out-of-sample data, accuracy measures and weighting scheme in a similar manner to `select_and_forecast`

```
R> mod1 <- midas_r(y ~ trend + mls(x, 4:14, 4, nealmon) +
R>                 mls(z, 12:22, 12, nealmon),
R>                 start = list(x = c(10, 1, -0.1), z = c(2, -0.1)))
R> avgf <- average_forecast(list(mod1),
R>                          data = list(y = y, x = x, z = z, trend = trend),
R>                          insample = 1:200, outsample = 201:250,
R>                          type = "fixed",
R>                          measures = c("MSE", "MAPE", "MASE"),
R>                          fweights = c("EW", "BICW", "MSFE", "DMSFE"))
```

It should also be pointed out that the forecast combinations in the function `select_and_forecast` are obtained only from the forecasts linked to different restriction functions on parameters. The forecasts related to different lag specifications are not combined, but the best lag order is chosen in terms of a given information criterion. If there is a need to get forecast combinations for a group of models which the user selected using other criteria, the function `average_forecast` should be used in a manner outlined in the previous example.

4. Empirical illustrations

4.1. Forecasting GDP growth

We replicate the example provided in Ghysels (2013). In particular we run MIDAS regression to forecast quarterly GDP growth with monthly non-farms payroll employment growth. The forecasting equation is the following

$$y_{t+1} = \alpha + \rho y_t + \sum_{j=0}^{8} \theta_j x_{3t-j} + \varepsilon_t,$$

where y_t is the log difference of quarterly seasonally adjusted real US GDP and x_{3t} is the log difference of monthly total employment non-farms payroll. The data is taken from St. Louis FRED website.

First we load the data and perform necessary transformations.

```
R> data("USqgdp")
R> data("USpayems")
R> y <- window(USqgdp, end = c(2011, 2))
R> x <- window(USpayems, end = c(2011, 7))
R> yg <- diff(log(y)) * 100
R> xg <- diff(log(x)) * 100
R> nx <- ts(c(NA, xg, NA, NA), start = start(x), frequency = 12)
R> ny <- ts(c(rep(NA, 33), yg, NA), start = start(x), frequency = 4)
```

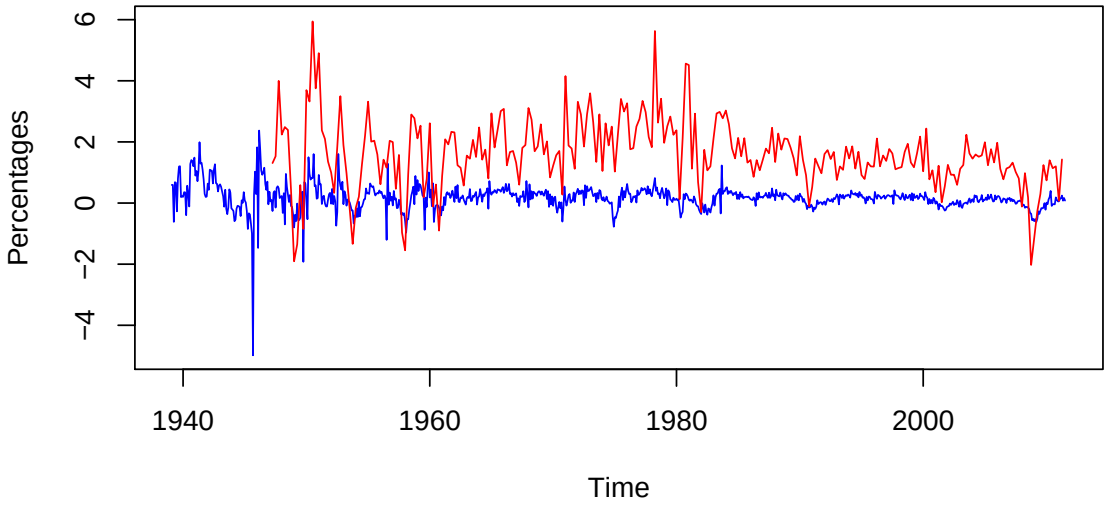

Figure 3: A plot of time series of quaterly gross domestic product growth rates and monthly non-farm payroll employment growth rates.

The last two lines are needed to equalize the sample sizes, which are different in the original data. We simply add additional `NA` values at the beginning and the end of the data. The graphical representation of the data is shown in Figure 3.

To specify the model for the `midas_r` function we rewrite it in the following equivalent form:

$$y_t = \alpha + \rho y_{t-1} + \sum_{j=3}^{11} \theta_j x_{3t-j} + \varepsilon_t,$$

As in Ghysels (2013) we restrict the estimation sample from the first quarter of 1985 to the first quarter of 2009. We evaluate the models with the Beta polynomial, Beta with non-zero and U-MIDAS weight specifications.

```
R> xx <- window(nx, start = c(1985, 1), end = c(2009, 3))
R> yy <- window(ny, start = c(1985, 1), end = c(2009, 1))
R> beta0 <- midas_r(yy ~ mls(yy, 1, 1) + mls(xx, 3:11, 3, nbeta),
+     start = list(xx = c(1.7, 1, 5)))
R> coef(beta0)

(Intercept)          yy         xx1         xx2         xx3
     0.8315      0.1059      2.5887      1.0201     13.6868

R> betan <- midas_r(yy ~ mls(yy, 1, 1) + mls(xx, 3:11, 3, nbetaMT),
+     start = list(xx = c(2, 1, 5, 0)))
R> coef(betan)
```

```
(Intercept)          yy         xx1         xx2         xx3         xx4
    0.93779     0.06748     2.26971     0.98659     1.49616    -0.09185

R> um <- midas_r(yy ~ mls(yy, 1, 1) + mls(xx, 3:11, 3), start = NULL)
R> coef(um)

(Intercept)          yy         xx1         xx2         xx3         xx4
    0.92990     0.08358     2.00047     0.88135     0.42965    -0.17597
        xx5         xx6         xx7         xx8         xx9
    0.28351     1.16285    -0.53082    -0.73392    -1.18732
```

We can evaluate the forecasting performance of these three models on the out of sample data, containing 9 quarters, from 2009Q2 to 2011Q2

```
R> fulldata <- list(xx = window(nx, start = c(1985, 1), end = c(2011, 6)),
+                   yy = window(ny, start = c(1985, 1), end = c(2011, 2)))
R> insample <- 1:length(yy)
R> outsample <- (1:length(fulldata$yy))[-insample]
R>
R> avgf<-average_forecast(list(beta0, betan, um),
+                         data = fulldata,
+                         insample = insample,
+                         outsample = outsample)
R> sqrt(avgf$accuracy$individual$MSE.out.of.sample)

[1] 0.5362 0.4767 0.4457
```

We see that the unrestricted MIDAS regression model gives the best out-of-sample RMSE.

4.2. Forecasting realized volatility

As another demonstration we use the package **midasr** to forecast the daily realized volatility. A simple model for forecasting the daily realized volatility was proposed by Corsi (2009). The heterogeneous autoregressive model of realized volatility (HAR-RV) is defined as

$$RV_{t+1}^{(d)} = c + \beta^{(d)} RV_t^{(d)} + \beta^{(w)} RV_t^{(w)} + \beta^{(m)} RV_t^{(m)} + w_{t+1},$$

where RV_t is the daily realized volatility and $RV_t^{(w)}$ and $RV_t^{(m)}$ are weekly and monthly averages:

$$RV_t^{(w)} = \frac{1}{5}\left(RV_t^{(d)} + RV_{t-1}^{(d)} + ... + RV_{t-4}^{(d)}\right)$$
$$RV_t^{(m)} = \frac{1}{20}\left(RV_t^{(d)} + RV_{t-1}^{(d)} + ... + RV_{t-19}^{(d)}\right),$$

where we assume that a week has 5 days, and a month has 4 weeks. This model is a special case of a MIDAS regression:

$$RV_{t+1}^{(d)} = c + \sum_{j=0}^{19} \beta_j RV_{t-j}^{(d)} + w_{t+1},$$

where

$$\beta_j = \begin{cases} \beta^{(d)} + \frac{1}{5}\beta^{(w)} + \frac{1}{20}\beta^{(m)}, \text{ for } j = 0, \\ \frac{1}{5}\beta^{(w)} + \frac{1}{20}\beta^{(m)}, \text{ for } j = 1, .., 4, \\ \frac{1}{20}\beta^{(m)}, \text{ for } j = 5, ..., 19. \end{cases}$$

The corresponding R code is the following

```
R> harstep <- function(p, d, m) {
+     if (d != 20)
+         stop("HAR(3)-RV process requires 20 lags")
+     out <- rep(0, 20)
+     out[1] <- p[1] + p[2]/5 + p[3]/20
+     out[2:5] <- p[2]/5 + p[3]/20
+     out[6:20] <- p[3]/20
+     out
+ }
```

For empirical demonstration we use the realized variance data on stock indices provided by Heber, Lunde, Shephard, and Sheppard (2009).

We estimate the model for the annualized realized volatility of the S&P500 index, which is based on 5-minute returns data.

```
R> data("rvsp500")
R> spx2_rvol <- 100 * sqrt(252 * as.numeric(rvsp500[, "SPX2.rv"]))
R> mh <- midas_r(rv ~ mls(rv, 1:20, 1, harstep), data = list(rv = spx2_rvol),
+     start = list(rv = c(1, 1, 1)))
R> summary(mh)


 Formula rv ~ mls(rv, 1:20, 1, harstep)

 Parameters:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   0.8304     0.3644    2.28  0.02273 *
rv1           0.3407     0.0446    7.63  3.0e-14 ***
rv2           0.4114     0.0693    5.93  3.2e-09 ***
rv3           0.1932     0.0508    3.80  0.00015 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

 Residual standard error: 5.56 on 3435 degrees of freedom
```

For comparison we also estimate the model with the normalized exponential Almon weights

```
R> mr <- midas_r(rv ~ mls(rv, 1:20, 1, nealmon), data = list(rv = spx2_rvol),
+     start = list(rv = c(0, 0, 0)), weight_gradients = list())
R> summary(mr)
```

```
 Formula rv ~ mls(rv, 1:20, 1, nealmon)

 Parameters:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   0.8377     0.3775    2.22    0.027 *
rv1           0.9447     0.0278   34.05  < 2e-16 ***
rv2          -0.7683     0.0961   -7.99  1.8e-15 ***
rv3           0.0291     0.0056    5.19  2.2e-07 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

 Residual standard error: 5.53 on 3435 degrees of freedom
```

We can test which of these restrictions is compatible with the data using the heteroscedasticity and autocorrelation robust weight specification test `hAhr_test`.

```
R> hAhr_test(mh)


        hAh restriction test (robust version)

data:
hAhr = 28.07, df = 17, p-value = 0.04408

R> hAhr_test(mr)


        hAh restriction test (robust version)

data:
hAhr = 19.27, df = 17, p-value = 0.3132
```

We can see that the null hypothesis pertaining to the HAR-RV-implied constraints in the MIDAS regression model is rejected at the 0.05 significance level, whereas the null hypothesis that the exponential Almon lag restriction is adequate, cannot be rejected.

Figure 4 illustrates the coefficients of the fitted MIDAS regressions and the coefficients of U-MIDAS regression with their corresponding 95% confidence bounds.

For the exponential Almon lag specification we can choose the number of lags via AIC or BIC.

```
R> tb <- expand_weights_lags("nealmon", from = 1, to = c(5, 15),
+                            start = list(nealmon = c(0, 0, 0)))
R> mtb <- midas_r_ic_table(rv ~ mls(rv, 1:20, 1, nealmon),
+                          data = list(rv = spx2_rvol), table = list(rv = tb),
+                          test = "hAh_test",
```

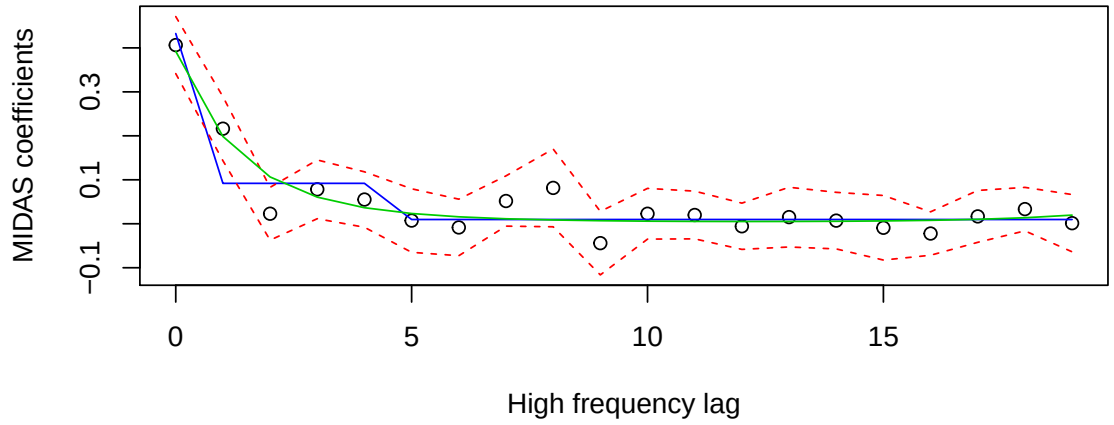

Figure 4: Comparison of HAR-RV (blue), Nealmon (green) and U-MIDAS (black) models.

```
+                          weight_gradients = list(), show_progress = FALSE)
R> mtb$candlist <- lapply(mtb$candlist, update, Ofunction = "nls")
R> mtb$test <- "hAhr_test"
R> mtb <- update(mtb)
```

Here we used two optimization methods to improve the convergence. The function `midas_r_ic_table` applies the `test` function for each candidate model. The function `hAhr_test` takes a lot of computing time, especialy for models with larger number of lags, so we calculate it only for the second final step, and we restrict the number of lags choose from. The AIC selects the model with 9 lags:

```
R> bm <- modsel(mtb)
```

```
 Selected model with AIC  =  21552
 Based on restricted MIDAS regression model
 The p-value for the null hypothesis of the test hAhr_test is 0.5532

 Formula rv ~ mls(rv, 1:9, 1, nealmon)

 Parameters:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)   0.9610     0.3694    2.60   0.0093 **
rv1           0.9371     0.0273   34.34  < 2e-16 ***
rv2          -1.1923     0.1929   -6.18  7.1e-10 ***
rv3           0.0966     0.0219    4.41  1.1e-05 ***
```

```
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

 Residual standard error: 5.52 on 3440 degrees of freedom
```

The HAC robust version of `hAh_test` again does not reject the null hypothesis of the exponential Almon lag specifications.

We can look into the forecast performance of both models, using a rolling forecast with 1000 observation window. For comparison we also calculate the forecasts of an unrestricted AR(20) model.

```
R> ar20 <- midas_r(rv ~ mls(rv, 1:20, 1), data = list(rv = spx2_rvol),
+                  start = NULL)
R> forc <- average_forecast(list(ar20, mh, bm),
+                            data = list(rv = spx2_rvol),
+                            insample = 1:1000, outsample = 1001:1100,
+                            type = "rolling", show_progress = FALSE)
R> forc$accuracy$individual
```

```
                          Model MSE.out.of.sample MAPE.out.of.sample
1          rv ~ mls(rv, 1:20, 1)             10.83              26.60
2 rv ~ mls(rv, 1:20, 1, harstep)             10.46              25.93
3  rv ~ mls(rv, 1:9, 1, nealmon)             10.35              25.90
  MASE.out.of.sample MSE.in.sample MAPE.in.sample MASE.in.sample
1             0.8200         28.62          21.57         0.8334
2             0.8020         29.25          21.59         0.8367
3             0.7945         29.08          21.81         0.8402
```

We see that exponential Almon lag model slightly outperforms the HAR-RV model and both models outperform the AR(20) model.

5. Final remarks

Only a part of the available functionality of the discussed functions of the package **midasr** was discussed. As it is usual in R, much more information on the resulting objects and all the information on the package-specific functions can be retrieved the using generic functions `objects` and `?`, respectively. Furthermore, in order to save space, the coding examples provided were almost always presented with minimal accompanying output obtained after running the code. The package page contains all the codes and complete output together with some additional illustration of the functionality of the package. Other information with a list of the functions and a number of demonstration codes is accessible using the usual `??midasr`.

6. Appendix

The Figure 1 was created using Monte-Carlo simulation. The following DGP was used

$$y_t = 2 + 0.1t + \sum_{j=0}^{16} \beta_j z_{12t-j} + u_t,\ z_\tau \sim N(0, \sigma^2),\ u_t \sim N(0, \sigma^2),$$

where z_τ and u_t are independent.

The coefficients β_j were chosen to come from the normalized exponential Almon polynomial restriction:

```
R> nealmon(p = c(2, 0.5, -0.1), d = 17)
```

The data for this DGP was generated for low frequency sample sizes 50, 100, 200, 300, 500, 750 and 1000. For each sample size an additional out-of-sample data set was generated using a quarter of the size of an in-sample data set. Three MIDAS regression models were estimated using in-sample data set: an unrestricted MIDAS, a restricted one using the correct constraint from the DGP, and the one with an incorrect restriction (non-exponential Almon polynomial). The forecast was calculated using the out-of-sample data-set. The Euclidean distance between the model coefficients and the coefficients of the DGP was recorded together with the mean squared error of the forecast.

This process was repeated 1000 times. The points in the figure are the averages of the replications. The code for generating the data can be found in the help page of a dataset `oos_prec` in the package **midasr**.

References

Andreou E, Ghysels E, Kourtellos A (2010). "Regression Models With Mixed Sampling Frequencies." *Journal of Econometrics*, **158**, 246–261.

Andreou E, Ghysels E, Kourtellos A (2011). "Forecasting with mixed-frequency data." In C M, D Hendry (eds.), *Oxford Handbook of Economic Forecasting*, pp. 225–245.

Andreou E, Ghysels E, Kourtellos A (2013). "Should macroeconomic forecasters look at daily financial data?" *Journal of Business and Economic Statistics*, **31**, 240–251.

Armesto M, Engemann K, Owyang M (2010). "Forecasting with mixed frequencies." *Federal Reserve Bank of St. Louis Review*, **92**, 521–536.

Bai J, Ghysels E, Wright J (2012). "State Space Models and MIDAS Regressions." *Econometric Reviews (forthcoming)*.

Bilinskas B, Kvedaras V, Zemlys V (2013). "Testing the functional constraints on parameters in cointegrated MIDAS regressions." Working paper, available upon a request.

Bilinskas B, Zemlys V (2013). "Testing the Functional Constraints on Parameters in Regression Models with Cointegrated Variables of Different Frequency." Submitted, available upon a request.

Breitung J, Roling C, Elengikal S (2013). "Forecasting inflation rates using daily data: A nonparametric MIDAS approach." Working paper, URL http://www.ect.uni-bonn.de/mitarbeiter/joerg-breitung/npmidas.

Clements M, Galvão A (2008). "Macroeconomic Forecasting with Mixed Frequency Data: Forecasting US output growth." *Journal of Business and Economic Statistics*, **26**, 546–554.

Corsi F (2009). "A simple approximate long-memory model of realized volatility." *Journal of Financial Econometrics*, **7**, 174–196.

Foroni C, Marcellino M, Schumacher C (2014). "U-MIDAS: MIDAS regressions with unrestricted lag polynomials." *Journal of the Royal Statistical Society âĂŞ Series A* (forthcoming).

Ghysels E (2013). "Matlab Toolbox for Mixed Sampling Frequency Data Analysis using MIDAS Regression Models." Available on Matlab Central at http://www.mathworks.com/matlabcentral/fileexchange/45150-midas-regression.

Ghysels E, Santa-Clara P, Valkanov R (2002). "The MIDAS touch: Mixed data sampling regression models." Working paper, UNC and UCLA.

Ghysels E, Santa-Clara P, Valkanov R (2006a). "Predicting volatility: getting the most out of return data sampled at different frequencies." *Journal of Econometrics*, **131**, 59–95.

Ghysels E, Sinko A, Valkanov R (2006b). "MIDAS regressions: Further results and new directions." *Econometric Reviews*, **26**, 53–90.

Ghysels E, Valkanov R (2012). "Forecasting volatility with MIDAS." In L Bauwens, C Hafner, S Laurent (eds.), *Handbook of Volatility Models and Their Applications*, pp. 383–401. Wiley.

Gilbert P, Varadhan R (2012). *numDeriv: Accurate Numerical Derivatives*. R package version 2012.9-1, URL http://CRAN.R-project.org/package=numDeriv.

Heber G, Lunde A, Shephard N, Sheppard K (2009). "Oxford-Man Institute's realized library." Oxford-Man Institute, University of Oxford. Library Version 0.2.

Kvedaras V, Račkauskas A (2010). "Regression Models with Variables of Different Frequencies: The Case of a Fixed Frequency Ratio." *Technical report, Oxford Bulletin of Economics and Statistics*, forthcoming.

Kvedaras V, Zemlys V (2012). "Testing the functional constraints on parameters in regressions with variables of different frequency." *Economics Letters*, **116**(2), 250–254.

Kvedaras V, Zemlys V (2013). "The statistical content and empirical testing of the MIDAS restrictions." Submitted, available upon a request.

Nash JC, Varadhan R (2011). "Unifying Optimization Algorithms to Aid Software System Users: optimx for R." *Journal of Statistical Software*, **43**(9), 1–14. URL http://www.jstatsoft.org/v43/i09/.

Rodriguez A, Puggioni G (2010). "Mixed frequency models: Bayesian approaches to estimation and prediction." *International Journal of Forecasting*, **26**, 293–311.

Wohlrabe K (2009). "Forecasting with mixed-frequency time series models." *Technical report*, Ph. D. Dissertation Ludwig-Maximilians-Universitat Munchen.

Zeileis A (2004). "Econometric Computing with HC and HAC Covariance Matrix Estimators." *Journal of Statistical Software*, **11**(10), 1–17. ISSN 1548-7660. URL http://www.jstatsoft.org/v11/i10.

Affiliation:

Eric Ghysels
Department of Economics
University of North Carolina - Chapel Hill
Gardner Hall, CB 3305 Chapel Hill, NC 27599-3305
E-mail: eghysels@unc.edu
URL: http://www.unc.edu/~eghysels/

Virmantas Kvedaras
Department of Econometric Analysis
Faculty of Mathematics and Informatics
Vilnius University
Naugarduko g. 24, Vilnius, Lithuania
E-mail: virmantas.kvedaras@mif.vu.lt
URL:http://mif.vu.lt/ututi/teacher/vk

Vaidotas Zemlys
Department of Econometric Analysis
Faculty of Mathematics and Informatics
Vilnius University
Naugarduko g. 24, Vilnius, Lithuania
E-mail: vaidotas.zemlys@mif.vu.lt
URL:http://mif.vu.lt/~zemlys/

Journal of Statistical Software http://www.jstatsoft.org/
published by the American Statistical Association http://www.amstat.org/
Volume VV, Issue II *Submitted:* yyyy-mm-dd
MMMMMM YYYY *Accepted:* yyyy-mm-dd