The ArfimaMLM Package for R

Version 1.3
(last update November 18, 2015)

Patrick Kraft[1] *Christopher Weber*[2] *Matthew Lebo*[1]

[1]Stony Brook University **[2]University of Arizona**

Abstract

`ArfimaMLM` provides functions to facilitate the estimation of Arfima-MLM for repeated cross-sectional data and pooled cross-sectional time series data. The estimation procedure uses double filtering with Arfima methods to account for autocorrelation in longer RCS followed by the use of multilevel modeling (MLM) to estimate both aggregate- and individual-level parameters simultaneously. This documentation provides a brief description of the general approach, as well as an introduction of possible model specifications. The main function of the package is `arfimaMLM`, which implements Arfima and multilevel models on a repeated cross-sectional dataset as described by Lebo and Weber (2015). Furthermore, the function `arfimaOLS` uses the same initial procedures but estimates a simple linear model instead of the multilevel model. The package also includes `arfimaPrep`, which prepares a dataset for subsequent analyses according to the Arfima-MLM framework without estimating the final model itself. `fd` is a wrapper function to estimate the fractional differencing parameter using `hurstSpec` of the `fractal`-package as well as procedures provided by the `fracdiff`-package (via ML, GPH, and Sperio) and to differentiate the series accordingly (mainly for internal use in `arfimaMLM`,`arfimaOLS`, and `arfimaPrep`).

Related Commands: `arfimaMLM`, `arfimaOLS`, `arfimaPrep`

---

1 Introduction

Lebo and Weber (2015) presented Arfima-MLM as a new framework for the analysis of repeated cross-sectional (RCS) data. The authors argued that previous methods either fail to account for possible autocorrelation in datasets with many time points, or do not allow for the simultaneous estimation of aggregate- and individual-level parameters. The method suggested by Lebo and Weber (2015) employs double filtering with Arfima methods to remove temporally deterministic components from the variables of interest. After purging potential autocorrelations, a multilevel model (MLM) is used to estimate the parameters for both, the individual as well as the (aggregate) temporal level variables.

The paper presented here describes the `ArfimaMLM`-package for the statistical software `R`, which allows for an easy implementation of the procedures described by Lebo and Weber (2015). The paper proceeds as follows: the first section will briefly summarize the statistical approach presented by Lebo and Weber (2015). The following section introduces the `ArfimaMLM`-package, describes the necessary steps for installation, and provides an overview over the general usage of the package. Subsequently, the package will be applied to a simulated dataset.

2 Arfima-MLM - Overview

Statistical analyses in political science are often based on data following a repeated cross-sectional designs. Common examples can be found in survey research, such as the National Annenberg Election Study, the American National Election Studies, or Gallup polls, or other types of data such as congressional roll-calls and Supreme Court cases (see Lebo and Weber 2015). Overall, repeated cross-sectional data structures can be characterized by the fact that individual observations are nested within time. However, in contrast to a panel structure, each unit is only included in the dataset at a single point in time. As Lebo and Weber (2015) point out, previous analyses that did not rely on multilevel models usually focused on either the aggregate- or the individual level when analyzing repeated cross-sectional data, which can lead to incorrect standard errors (in the case of pooling) or a significant information loss (in the case of aggregating over time points). Multilevel modeling overcomes this dichotomy between individual and aggregate perspectives since it allows for the estimation of parameters on both levels. However, a simple multilevel model that does not take into account potential deterministic components on the aggregate level might still yield biased or inconsistent results.

The estimation procedure suggested by Lebo and Weber (2015) combines established approaches used to analyze long time series and multilevel modeling in order to account for potential autocorrelations on the aggregate level.

The first step of the Arfima-MLM approach consists of the estimation of fractional differencing parameters for the aggregate-level variables considered in the analyses and differencing the respective series accordingly (see also Box-Steffensmeier and Smith 1996), such that

$$\bar{Y}_t^* = (1 - L)^d \bar{Y}_t, \tag{1}$$

where $\bar{Y}_t^*$ is the stationary series free of autocorrelation, L is a lag operator, d is the fractional differencing parameter, and $\bar{Y}_t$ is the original series of level means of Y at time t.

Furthermore, the procedure described by Lebo and Weber (2015) includes the estimation of AR and MA parameters as part of the Arfima (p,d,q) model. The fractional differencing and modeling of AR and MA parameters described above is employed for the dependent variable as well as all remaining independent aggregate-level variables included in the analyses (see Lebo and Weber 2015 for a more detailed discussion of the approach).

In the following step, the individual-level variables are purged from serial correlation on the aggregate level. For the dependent variable, this is done by subtracting the daily deterministic component ($\bar{Y}_t - \bar{Y}_t^*$) from the individual-level values:

$$y_{it}^{**} = y_{it} - (\bar{Y}_t - \bar{Y}_t^*), \tag{2}$$

where y_{it}^{**} consists of the within-time point, as well as the white-noise between-time point variation.

In order to remove potential autocorrelation from independent (individual-level) variables, the within-time point variation is calculated by subtracting the respective level means at time t,

$$x_{it}^{**} = x_{it} - \bar{X}_t, \tag{3}$$

where x_{it}^{**} is the within-time point variation of x_{it}, x_{it} is the original individual-level variable, and $\bar{X}_t$ is the variable's level mean at time t.

After applying this double-filtering procedure to the variables of interest, the multilevel model can be specified. Consider an example with a dependent variable y_{it}, an independent variable x_{it} (both of which are filtered according to the procedure outlined above), as well as a aggregate-level variable Z (which does not vary within time-points). The respective model can be described as follows (Lebo and Weber, 2015):

$$y_{it}^{**} = \alpha_{1t} + \beta_1 x_{it}^{**} + \epsilon_{1it} \tag{4}$$

$$\alpha_{1t} = \alpha_{2t} + \beta_2 \bar{X}_t^* + \gamma Z_t^* + \epsilon_{2t}, \tag{5}$$

where y_{it}^{**} are the double filtered values for y_{it}, x_{it}^{**} are the within-time point variations of x_{it}, $\bar{X}_t^*$ is the fractionally differenced series of $\bar{X}_t$, and Z_t^* is the fractionally differenced series of Z_t. As described by Lebo and Weber (2015), equation (4) can be further expanded to incorporate time-varying coefficients at the individual level, e.g. by letting the coefficient for x_{it}^{**} vary across time points:

$$y_{it}^{**} = \alpha_{1t} + \beta_t x_{it}^{**} + \epsilon_{1it} \tag{6}$$

The following section describes the `ArfimaMLM`-package which facilitates the estimation based on the procedure outlined above.

3 Installing the Package

`ArfimaMLM` is available on CRAN and can be installed by executing the following command in `R`:

```
install.packages("ArfimaMLM")
```

Alternatively, the latest (development) version of `ArfimaMLM` can be installed directly from GitHub (using the `devtools`-Package):

```
library(devtools)
install_github("pwkraft/ArfimaMLM")
```

Now the package is installed to the library. In every new session, the package can now be loaded by executing

```
library(ArfimaMLM)
```

At the current version (1.3), the package consists of three major functions: `arfimaMLM`, `arfimaOLS`, and `arfimaPrep`, as well as `fd`, which is mainly for internal use within the remaining functions.[1] `arfimaMLM` employs the data manipulations and analyses for a given repeated cross-sectional dataset as specified above. `ArfimaOLS` performs the same data manipulations but ultimately estimates a linear model instead of a multilevel model. `arfimaPrep` allows the user to manipulate the data according to the framework specified above, but without estimating a final model. `fd` is a wrapper function for the `fracdiff`-package for internal use within `arfimaPrep`. The following section will provide an introduction for the usage of the `arfimaMLM` command on the basis of a simulated dataset.

4 Using the arfimaMLM-command - A Brief Example

4.1 Simulating a Repeated Cross-Sectional Dataset

In order to demonstrate the usage of the `arfimaMLM` command, we constructed a simulational scenario of a repeated cross-sectional dataset with 100 time points and 500 units within each time point. The dataset contains four different independent variables: x_1, x_2, z_1, and Z_2. x_1 is normally distributed with mean $\bar{X}_{1t}$ and a standard deviation of 2. Across time points t, $\bar{X}_{1t}$ follows a fractionally integrated series with $d = 0.3$ and a mean of 5. x_2 is normally distributed with a mean of 0 and a standard deviation of 40. The mean of x_2 is constant across all time points. z_1 is constructed similar to x_1: the variable is normally distributed with a mean of $\bar{Z}_{1t}$ and a standard deviation of 3. $\bar{Z}_{1t}$ follows a fractionally integrated series with $d = 0.1$ and a mean of 2. Z_{2t} follows a fractionally integrated series with $d = 0.25$ and a mean of 3. Z_{2t} does not differ within time points. The dependent variable y is constructed as follows:

$$\begin{aligned} y &= \bar{Y}_t + \beta_{1t} * x_1 - 0.05 * x_2 + 0.3 * \bar{Z}_{1t} + 0 * Z_{2t} + \epsilon \qquad \text{, where} \\ \beta_{1t} &\sim N(0.2, \sigma^2 = 0.1) \\ \epsilon &\sim N(0, \sigma^2 = 1), \end{aligned} \tag{7}$$

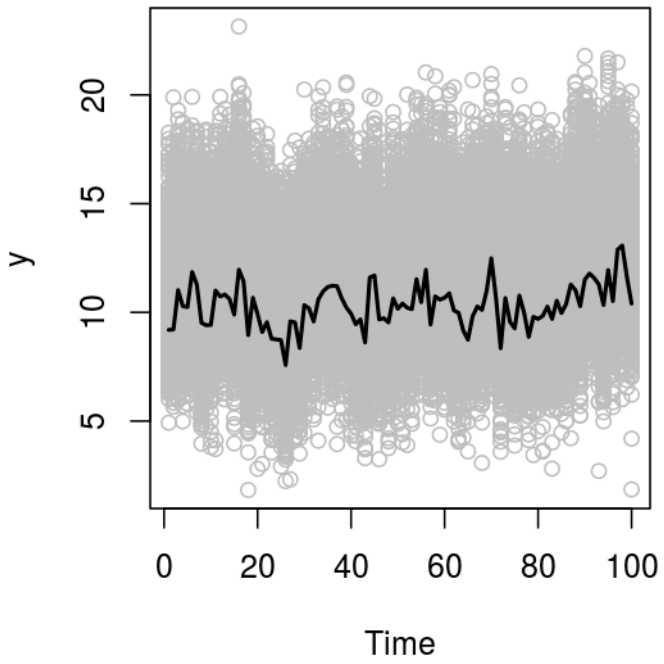
where $\bar{Y}_t$ follows a fractionally integrated series with $d = 0.4$ and a mean of 10. Note that β_{1t} is a random coefficient that varies between time points and that the level variable Z_{2t}

[1] For details, execute `?arfimaMLM`, `?arfimaOLS`, `?arfimaPrep`, or `?fd` in `R`, or refer to the package manual.

actually has *no* effect on y. In the dataset, we observe the the following variables: y, x_1, x_2, and z_1 for each individual i, as well as Z_{2t} for each time point t. It is worth noting that in this scenario, it is the aggregate level at time t of z_1 that affects y, while we measure the variable on an individual level. This aspect was included to demonstrate the possibility of testing aggregate level hypotheses based on individual level data within the `arfimaMLM` package without prior data manipulations.

The following figures provide an overview over the data simulated for further analyses. Figure 1 displays the individual level data points for y as well as the underlying fractionally integrated mean $\bar{Y}_t$.

Fig. 1: Plot of Dependent Variable y Across Time

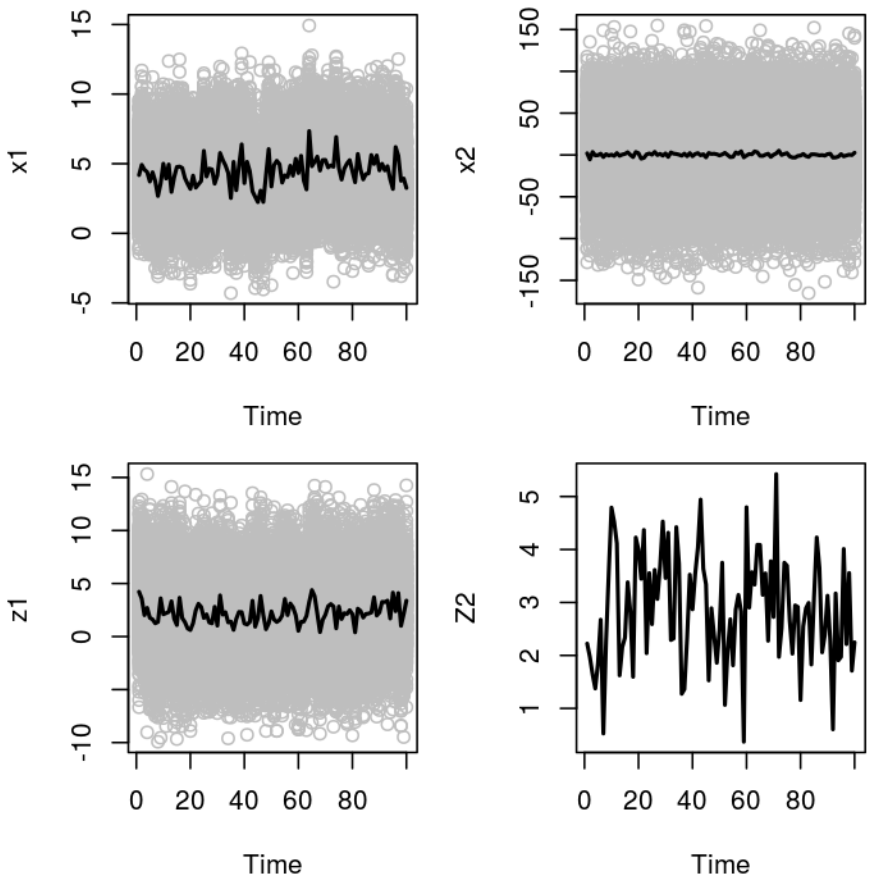


Unsurprisingly, it can be observed that the fractionally integrated structure of the level means manifests itself in the central tendencies of the individual variable values at time t. Figure 2 displays the same plots for the four independent variables.

Both individual level variables that were specified to follow an underlying fractionally integrated mean (x_1 and z_1) show the same pattern as the dependent variable y. As specified in the model, the mean of x_2 does not vary across time points. For Z_2, no individual level variables are observed, so the plot only displays the respective level means at time t.

After presenting the simulational scenario, we will now turn to the discussion of the model estimation.

Fig. 2: Plot of Independent Variables Across Time



4.2 Model Estimation

As already outlined above, the dataset only contains the following variables x_1, x_2, z_1, Z_2 and y. Calling the head of the data frame in `R` yields the following output:

```
head(data)

  time        x1        x2         z1       z2         y
1    1 4.0377894 -32.12057  6.4554502 2.222936 12.250030
2    1 1.8475985 -98.35277  6.5237611 2.222936 15.278788
3    1 2.9154049  33.02954 -1.7316546 2.222936  8.544409
4    1 4.1272183  95.09765  2.3748715 2.222936  6.813665
5    1 5.5262933 -18.80429  6.1095797 2.222936 11.634492
6    1 0.8838083 -10.80839  0.7090984 2.222936 10.483916
```

It is worth noting again that x_1, x_2, and y are only included as individual-level variables. z_1 is also included as an individual-level variable but we are ultimately interested in the estimation of the level effect of $\bar{Z}_t$. Z_2 on the other hand does not vary between time points.

In order to be able to compare the model estimates of `arfimaMLM` to other model specification, we will first present the results of a simple linear regression, as well as a multi-level model (including random effects for `x1`) based on the simulated data. The results are presented in Table 1.

Tab. 1: Results for Simple OLS Model and Multilevel Model

	Dependent variable:	
	y	
	OLS	*linear mixed-effects*
	(1)	(2)
x1	0.200***	0.207***
	(0.003)	(0.011)
x2	−0.050***	−0.050***
	(0.0002)	(0.0001)
z1	0.025***	0.003*
	(0.002)	(0.001)
z2	−0.223***	−0.181*
	(0.006)	(0.099)
Constant	11.546***	11.455***
	(0.024)	(0.304)
Observations	50,000	50,000
Note:	*p<0.1; **p<0.05; ***p<0.01	

While most of the estimates seem to be reasonably close to the scenario specification, it should be noted that the coefficient for Z_2 is highly significant even though it was specified to actually have a null effect. It is very likely that this result arises due to the fact that we did not account for autocorrelations in the series of Z_2 or y, which potentially induced spurious correlations between both variables. Furthermore, the estimated coefficient for Z_1 appears to be much smaller than the true value, albeit still being significant.

How can the model be estimated with the `arfimaMLM` package? The basic structure of the command can be described as follows:

```
arfimaMLM(formula, data, timevar
          , d = "Hurst", arma = NULL
          , ecmformula = NULL, decm = "Hurst"
          , drop = 5, report.data = TRUE, ...)
```

The major input for the function is `formula`, which specifies the multilevel model to be estimated after the necessary data transformations according to the Arfima-MLM framework were implemented for the dataset specified in `data`. It can be specified similar to the model

call in standard multilevel models estimated via `lmer`. The only difference is that the variables specified in `formula` do not have to be equal to the original variable names included in the data frame. Rather, the user can add specific suffixes to each variable in order to call specific data transformations in the `arfimaMLM` function. In the current version of the package, three specific suffixes are possible: ".ydif", ".xdif" and ".fd".

If the suffix `.ydif` is added to the dependent variable (e.g. `y.ydif` instead of `y`), the function will remove the daily deterministic component from the individual level variable as specified in equation (2), such that it only consists of within-time point, as well as non-temporally autocorrelated between-time point variation. If the suffix `.xdif` is added to an independent variable (e.g. `x1.xdif` instead of `x1`), the variable is simply filtered through the time point averages as specified in equation (3).

The suffix `.fd` allows the user to select variables which are supposed to be transformed to a fractionally differenced level-variable (by aggregating individuals over each time point prior to fractionally differencing the series, `z1` in our example), or variables which are already included as a level-variable in the original dataset and are just supposed to be fractionally differenced before the multilevel model is estimated (`z2` in our example). Since the suffixes `.ydif`, `.xdif`, and `.fd` are interpreted by the function as calls for specific data transformation procedures, none of the original variable names in `data` should include them in order to prevent errors in the estimation procedure.

Consider the following examples for possible `formula` calls in the `arfimaMLM` package (omitting the remaining arguments) for the dataset specified above:

```
arfimaMLM(y.ydif ~ x1.xdif + x2 + z1.fd + z2.fd
          + (1|time)
          , data=data, timevar = "time", ...)
arfimaMLM(y.ydif ~ x1.xdif + x2 + z1.fd + z2.fd
          + (1+x1.xdif|time)
          , data=data, timevar = "time", ...)
```

For the simulational scenario, we want to filter the variables `y` and `x1`. Accordingly, the suffix `.ydif` is added to the name of the dependent variable, and the suffix `.xdif` is added to the name of the independent variable in the data frame `data`. Furthermore, we want to include the level effects of `z1` and `z2`. In order to aggregate `z1` and fractionally difference `z1` and `z2`, we add the suffix `.fd`. The level variable for these transformations is specified by including the respective variable name `time` as the argument for `timevar`. Furthermore, this variable is also specified in `formula` as the variable according to which the observations are nested in the multilevel model. In the second `arfimaMLM`, we additionally include a random coefficient for `x1.xdif`.

We can also add an error correction mechanism to the model. Note that the ecm again does not have to be part of the original dataset but rather is generated within the function. The following two examples show how the error correction mechanism can be specified as part of the `arfimaMLM` procedure.

```
arfimaMLM(y.ydif ~ x1.xdif + x2 + z1.fd + z2.fd + ecm
          + (1|time)
          , data=data, timevar = "time"
          , ecmformula = y.mean ~ x1.mean, ...)
arfimaMLM(y.ydif ~ x1.xdif + x2 + z1.fd + z2.fd + ecm
          + (1+x1.xdif|time)
          , data=data, timevar = "time"
          , ecmformula = y.mean ~ x1.mean, ...)
```

`ecmformula` contains the specification of the co-integration regression to receive the residuals for the error correction mechanism (`ecm`) included in `formula` according to a simple linear regression model specification. Note that the variable names included here cannot be the original variable names, but rather have to be supplemented by the suffix `.mean`, since the ecm is based on the level/time aggregates at time t. Again, the mean is calculated based on the level variable specified in `timevar`, namely `time`.

If `formula` contains `ecm` as one of the independent variables, and `ecmformula` is correctly specified, the function will include the lag of the fractionally differenced residuals of the co-integration regression as an error correction mechanism in the multilevel model. The ECM does not have to be estimated prior to calling the function.

The remaining arguments entered in the function specify details about the data manipulation. `d` calls for a specific estimation method for the fractional differencing parameter in the `fracdiff`-package or the `fractal`-package (`"Hurst"`, `"ML"`, `"GPH"`, or `"Sperio"`). The default is `"Hurst"`. If the user wants to specify the methods for each variable individually, `d` can be a list containing a call for every individual variable. Furthermore, the list can contain numeric values for differencing parameters which were estimated externally (see example). A variable will not be differenced if `d` is specified as 0. Accordingly, `decm` calls for a specific estimation method for the fractional differencing parameter. Again, it can be either `"Hurst"`, `"ML"`, `"GPH"`, or `"Sperio"` and the default is `"Hurst"`. Again, the argument can also be a numeric value indicating the differencing parameter estimated externally.

It is also possible to estimate AR and MA parameters for the (fractionally differenced) level variables in order to remove remaining autocorrelation. `arma` contains a list of variables for which AR and MA parameters are to be estimated (after fractional differencing) as well as a vector containing the respective orders of the model to fit. `order[1]` corresponds to the AR part and `order[2]` to the MA part, similar to the model specification in `arima` (just excluding the d parameter here). For variables specified in `arma`, the function will use the residuals of the ARMA model (which is estimated for the fractionally differenced level variables, respectively) for the subsequent model estimation in order to remove their deterministic components. This procedure is only available for variables which were augmented by the suffix `.fd` or `.ydif` in `formula`. It is also possible to fix certain AR or MA parameters at zero instead of estimating all parameters up to the order described in `arma`. For example, one might want to estimate AR(1) and AR(3) parameters, but not include AR(2) for a specific variable in the model.

Consider the following examples for different model specifications taking into account the estimation method for the fractional differencing parameter as well as the estimation of

respective ARMA models.

```
arfimaMLM(y.ydif ~ x1.xdif + x2 + z1.fd + z2.fd + ecm
          + (1|time)
          , data=data, timevar = "time"
          , ecmformula = y.mean ~ x1.mean
          , d="Sperio"
          , decm="ML", ...)
arfimaMLM(y.ydif ~ x1.xdif + x2 + z1.fd + z2.fd + ecm
          + (1+x1.xdif|time)
          , data=data, timevar = "time"
          , ecmformula = y.mean ~ x1.mean
          , d=list(y="Hurst", z1="Sperio", z2=0.25)
          , decm="GPH", ...)
arfimaMLM(y.ydif ~ x1.xdif + x2 + z1.fd + z2.fd + ecm
          + (1+x1.xdif|time)
          , data=data, timevar = "time"
          , ecmformula = y.mean ~ x1.mean
          , arma = list(y = c(1,0), z2 = c(0,2)), ...)
arfimaMLM(y.ydif ~ x1.xdif + x2 + z1.fd + z2.fd + ecm
          + (1+x1.xdif|time)
          , data=data, timevar = "time"
          , ecmformula = y.mean ~ x1.mean
          , arma = list(y = list(1,c(1,3))
                        , z2 = c(0,1)))
```

The first example calls the estimation method `Sperio` for all respective variables included in `formula`, and `ML` for fractionally differencing the error correction mechanism. In the second example, `d` contains a list specifying the estimation method for each variable individually. Accordingly, `y` is estimated via `Hurst`, and `z1` via `Sperio`. For `z2`, the fractional differencing parameter was externally set to 0.25. The fractional integration parameter for the error correction mechanism is estimated via `GPH`. While it is unlikely that such a detailed specification of estimation mechanisms is necessary (or useful), this example was merely supposed to demonstrate the flexibility of the function. In the third example, the estimation of the d parameters for all respective variables in `formula` and `ecm` is kept at the default (`Hurst`). However, additionally to fractional differencing, the function also estimates an AR(1) model for `y` and a MA(2) model for `z2`. Note that the MA(2) model will provide estimates for both, MA(1) and MA(2). The multilevel model estimated subsequently would include the residuals of the AR/MA model estimated after fractionally differencing the respective variables. The last example presents a more complex ARMA model specification for `y` than in the previous example. The only difference is, that the argument indicating the order of the ARMA model for `y` is a list (`list(1,c(1,3))`) instead of a vector (`c(1,0)`). This implies that the ARMA model does not include all parameters up to the highest order specified (as in the previous example), but rather only include the AR and MA parameters that are explicitly listed. Again, the first element corresponds to the AR part, while the

second element corresponds to the MA part of the model. Accordingly, `arma = list(y = list(1,c(1,3)))` will estimate AR(1), MA(1) and MA(3) parameters for y, while `arma = list(y = c(1,3))` would yield estimates for AR(1), MA(1), MA(2), as well as MA(3).

The last two arguments that can be specified are `drop`, which determines the number of initial time points to be dropped from the analysis (default is 5), as well as `report.data`, which determines whether the model output should include the transformed dataset used for the analysis (default is `TRUE`). Furthermore, additional arguments can be passed to the estimation procedures used within the function (e.g. for `lmer`).

After describing the details about the model specification, we will now turn to a brief discussion of the function's output. In general, the function returns a list of the S3 class `'arfimaMLM'` with the following items:

- `result`: Output of the multilevel model as specified in `formula`.
- `ecm`: Output of the co-integration regression (returned if `ecmformula` is specified). The lagged residuals of the co-integration regression are included in the multilevel model if `ecm` is included in `formula`.
- `d`: Matrix of fractional differencing parameters estimated for the level variables (`.ydif` and `.fd`) as well as the estimation method for each variable. Returns the specified value for d if it was specified in the initial call of the function.
- `arma`: List of `arima` results for each variable specified in the model call. Contains AR/MA estimates as well as the model residuals.
- `data.mean`: Data frame of variable means declared in `formula` as `.ydif`, `.xdif` or `.fd` for each time point specified by the level variable in `formula`.
- `data.fd`: Data frame of fractionally differenced level variables declared in `formula` as `.ydif` or `.fd` for each time point specified by the level variable in `formula`.
- `data.merged`: Merged data frame used when estimating the multilevel model consisting of the original data as well as `data.mean` and `data.fd`

Note that `data.mean`, `data.fd`, and `data.merged` are only included if `report.data` is set at its default (i.e. `TRUE`). Consider the following example for the simulational scenario specified in this paper:

```
m1 <- arfimaMLM(y.ydif ~ x1.xdif + x2 + z1.fd + z2.fd
                + ecm + (1+x1.xdif|time)
                , data = data, timevar = "time"
                , ecmformula = y.mean ~ x1.mean
                , d = "Hurst"
                , decm = "GPH")
```

Calling the summary of this model will provide a brief R-Output that summarizes the most important results:

```
summary(m1)
```

```
###################################
Summary Error Correction Mechanism:

Call:
lm(formula = ecmformula, data = data.mean)

Residuals:
    Min      1Q  Median      3Q     Max
-3.2427 -0.7130  0.0182  0.7618  2.6427

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  11.1164     0.5400  20.584   <2e-16 ***
x1.mean       0.1630     0.1194   1.364    0.176
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.13 on 98 degrees of freedom
Multiple R-squared:  0.01864,Adjusted R-squared:  0.008626
F-statistic: 1.861 on 1 and 98 DF,  p-value: 0.1756


###################################
Fractional Differencing Parameters:

    Method   Estimate
y    Hurst 0.41713434
z1   Hurst 0.01679005
z2   Hurst 0.10887414
ecm    GPH 0.55248930


#########################
Summary Multilevel Model:

Linear mixed model fit by REML ['lmerMod']
Formula: y.ydif ~ x1.xdif + x2 + z1.fd + z2.fd + ecm + (1 + x1.xdif |
    time)
   Data: new$data.merged

REML criterion at convergence: 136183.4

Scaled residuals:
```

```
    Min      1Q  Median      3Q     Max
-4.2365 -0.6678  0.0050  0.6697  4.0721

Random effects:
 Groups   Name        Variance Std.Dev. Corr
 time     (Intercept) 1.11753  1.057
          x1.xdif     0.01167  0.108    0.47
 Residual             1.01005  1.005
Number of obs: 47500, groups:  time, 95

Fixed effects:
              Estimate Std. Error t value
(Intercept)  0.0600844  0.1087158     0.6
x1.xdif      0.2035655  0.0113217    18.0
x2          -0.0499160  0.0001156  -431.7
z1.fd        0.2360076  0.1037031     2.3
z2.fd       -0.0468624  0.0943287    -0.5
ecm         -0.1497914  0.0873453    -1.7

Correlation of Fixed Effects:
        (Intr) x1.xdf x2     z1.fd  z2.fd
x1.xdif  0.455
x2       0.000  0.000
z1.fd    0.030  0.000  0.000
z2.fd   -0.036  0.000 -0.001  0.005
ecm     -0.030  0.000  0.000 -0.021  0.077
```

Looking at the coefficients estimated in the multilevel model of the `arfimaMLM` function, it can be seen that the coefficients are closer to the values specified in the simulational scenario as compared to the estimates reported for the simple OLS model (see Table 1). More specifically, the estimate for `z1` is closer to its true value. More importantly, taking into account the time-series structure of the level-variables eliminates the significant effect of `z2`. While this evidence by itself is obviously not sufficient to show that the ArfimaMLM procedure yields more consistent estimates in general, it certainly provides additional evidence in favor of the findings presented by Lebo and Weber (2015).

We can also extract specific items from the `arfimaMLM`-list. Let's assume we were only interested in the result of the multilevel model. Furthermore, we would like to take a look at the head of the fractionally differenced data as well as the data containing the generated means for each time point. We would also like to save the merged data frame used for the analysis for future use. All of that can be implemented by executing the following code:

```
summary(m1$result)

Linear mixed model fit by REML ['lmerMod']
Formula: y.ydif ~ x1.xdif + x2 + z1.fd + z2.fd + ecm + (1 + x1.xdif |
```

```
    time)
   Data: new$data.merged

REML criterion at convergence: 136183.4

Scaled residuals:
    Min      1Q  Median      3Q     Max
-4.2365 -0.6678  0.0050  0.6697  4.0721

Random effects:
 Groups   Name        Variance Std.Dev. Corr
 time     (Intercept) 1.11753  1.057
          x1.xdif     0.01167  0.108    0.47
 Residual             1.01005  1.005
Number of obs: 47500, groups:  time, 95

Fixed effects:
             Estimate Std. Error t value
(Intercept)  0.0600844  0.1087158     0.6
x1.xdif      0.2035655  0.0113217    18.0
x2          -0.0499160  0.0001156  -431.7
z1.fd        0.2360076  0.1037031     2.3
z2.fd       -0.0468624  0.0943287    -0.5
ecm         -0.1497914  0.0873453    -1.7

Correlation of Fixed Effects:
        (Intr) x1.xdf x2     z1.fd  z2.fd
x1.xdif  0.455
x2       0.000  0.000
z1.fd    0.030  0.000  0.000
z2.fd   -0.036  0.000 -0.001  0.005
ecm     -0.030  0.000  0.000 -0.021  0.077

head(m1$data.fd)

  time       y.fd      z1.fd       z2.fd        ecm
1    1 -0.4980316  1.9757137 -0.65226663         NA
2    2  0.4986830  1.3185836 -0.83418872 -0.4669326
3    3  0.7585310 -0.1471850 -1.10178838  0.4668722
4    4 -0.1486040  0.3205121 -1.30089067  0.7160995
5    5 -0.2049091 -0.2496606 -0.75823137 -0.2124583
6    6  0.3718751 -0.5550096  0.05193581 -0.1185358

head(m1$data.mean)
```

```
  time   y.mean  x1.mean  z1.mean  z2.mean
1    1 11.33888 4.230249 4.154311 2.222936
2    2 12.12785 4.924501 3.530353 1.969999
3    3 12.65626 4.700633 2.070416 1.643219
4    4 12.03351 4.320279 2.519231 1.376323
5    5 11.81165 3.712363 1.949179 1.851921
6    6 12.27197 4.375128 1.633863 2.674118

newdata <- m1$data.merged
```

The same model can also be estimated using `arfimaOLS`. All arguments with regard to the data manipulations and model specification are equivalent to `arfimaMLM`. The only difference is that `arfimaOLS` ultimately estimates a simple linear regression model rather than a multilevel model. Consider the following specification of the same model using `arfimaOLS`:

```
m2 <- arfimaOLS(y.ydif ~ x1.xdif + x2 + z1.fd + z2.fd
                + ecm
                , data = data, timevar = "time"
                , ecmformula = y.mean ~ x1.mean
                , d = "Hurst"
                , decm = "GPH")
```

The only difference in the function call for `arfimaOLS` is the fact that `formula` does not include the specification for clustering. All other arguments, function calls, as well as the structure of the output is equivalent to the description outlined above.[2] Again, calling the summary of this model will provide a brief R-Output that summarizes the most important results:

```
summary(m2)

###################################
Summary Error Correction Mechanism:

Call:
lm(formula = ecmformula, data = data.mean)

Residuals:
    Min      1Q  Median      3Q     Max
-3.2427 -0.7130  0.0182  0.7618  2.6427

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
```

[2] It should be noted that the S3 class of this model output is `arfimaOLS` and not `arfimaMLM`. However, both classes are essentially equivalent with regard to their structure and behavior.

```
(Intercept)  11.1164     0.5400  20.584   <2e-16 ***
x1.mean       0.1630     0.1194   1.364    0.176
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.13 on 98 degrees of freedom
Multiple R-squared:  0.01864,Adjusted R-squared:  0.008626
F-statistic: 1.861 on 1 and 98 DF,  p-value: 0.1756


###################################
Fractional Differencing Parameters:

    Method   Estimate
y    Hurst 0.41713434
z1   Hurst 0.01679005
z2   Hurst 0.10887414
ecm    GPH 0.55248930


##################
Summary OLS Model:

Call:
lm(formula = formula, data = new$data.merged)

Residuals:
    Min      1Q  Median      3Q     Max
-6.0890 -0.9588  0.0402  1.0017  5.7285

Coefficients:
              Estimate Std. Error  t value Pr(>|t|)
(Intercept)  0.0613607  0.0067106    9.144   <2e-16 ***
x1.xdif      0.2033355  0.0033485   60.724   <2e-16 ***
x2          -0.0499045  0.0001677 -297.662   <2e-16 ***
z1.fd        0.2076489  0.0071888   28.885   <2e-16 ***
z2.fd       -0.0846972  0.0065389  -12.953   <2e-16 ***
ecm         -0.1696242  0.0060560  -28.009   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.46 on 47494 degrees of freedom
Multiple R-squared:  0.6641,Adjusted R-squared:  0.664
F-statistic: 1.878e+04 on 5 and 47494 DF,  p-value: < 2.2e-16
```

Again, the estimate for `z1` appears to be closer to its true value as compared to the original OLS results. However, since we did not properly take into account the multilevel structure as in `arfimaMLM`, the coefficient for `z2` still remains significant (albeit being closer to zero than the coefficient reported in Table 1).

References

Box-Steffensmeier, Janet M and Renee M Smith. 1996. "The dynamics of aggregate partisanship." *American Political Science Review* 90(3):567–580.

Lebo, Matthew and Christopher Weber. 2015. "An Effective Approach to the Rolling Cross Sectional Design." *American Journal of Political Science* 59(1):242–258.