

# The MAR package

Andrea Bucci

Giulio Palomba

Marco Tedeschi

version 1.0

## Abstract

This document covers version 1.0 of the **gretl** MAR package. The very first version of the package (version 0.1) was presented at the **gretl** conference in Birmingham in June 2025, and is covered in detail in the paper by Bucci, Palomba, and Tedeschi [2026]. In practice, the following pages are intended to illustrate the differences compared to the previous version.

## 1 Introducing the constant

The difference between version 0.1 and the current version 1.0 consists in the possibility of estimating the MAR(1) model by inserting the constant, therefore all the functions, both public and default, have been modified taking into account the possible presence of this constant matrix. Accordingly, the following sections show how inserting a constant into MAR(1) improves the model from an analytical point of view and leads to some necessary changes within the package functions. It is worth noting that the inclusion of a constant matrix in the model is not considered in the seminal paper of Chen, Xiao, and Yang [2021], so this guide can be seen as an attempt to generalize their approach.

The reminder of this guide proceeds as follows. Sections 1.1 and 1.2 provide some analytics regarding the definition and the estimation of the MAR(1) model once a matrix of constants is inserted. Section 1.3 describes in short how the package functions have been updated after the introduction of such matrix, while section 1.4 shows the GUI. Section 2 illustrates the example file contained in the MAR package and finally section 3 provides a sort of historical overview of the package by presenting the main characteristics and features of its different versions.

### 1.1 MAR(1) model with constant

As mentioned above, version 1.0 of the MAR package allows one to estimate a matrix containing the intercepts. Consequently, equation (3) in Bucci, Palomba, and Tedeschi [2026, hereafter COMPSTAT] becomes

$$\mathbf{Y}_t = \boldsymbol{\mu} + \mathbf{A}\mathbf{Y}_{t-1}\mathbf{B}' + \mathbf{E}_t, \quad (1)$$

where  $\mathbf{Y}_t$  is the  $m \times n$  matrix containing data,  $\boldsymbol{\mu}$  is the  $m \times n$  matrix of constants,  $\mathbf{A}$  is a square  $m$ -dimensional parameter matrix with unit Frobenius norm, and  $\mathbf{B}$  is a square  $n$ -dimensional parameter matrices. The error has some distribution with  $E(\mathbf{E}_t) = \mathbf{0}$  and covariance matrix

$$\underset{(mn \times mn)}{\boldsymbol{\Sigma}} = \boldsymbol{\Sigma}_c \otimes \boldsymbol{\Sigma}_r, \quad (2)$$

where

$$\underset{(n \times n)}{\boldsymbol{\Sigma}_c} = \frac{1}{m(T-p)} \sum_{t=p+1}^T \mathbf{E}_t' \mathbf{E}_t \quad \text{and} \quad \underset{(m \times m)}{\boldsymbol{\Sigma}_r} = \frac{1}{n(T-p)} \sum_{t=p+1}^T \mathbf{E}_t \mathbf{E}_t'. \quad (3)$$

The model (1) contains  $m^2 + n^2 + mn$  parameters to be estimated.

## 1.2 Estimation

Since the available estimation models are iterated least squares (LSE) and Maximum Likelihood (MLE), in the former case the optimization problem is

$$\min_{\Theta} \|Y_t - \mu - AY_{t-1}B'\|_F^2$$

where  $\Theta = \{\mu, A, B\}$  and  $\|\cdot\|_F^2$  denotes the Frobenius norm. The latter case is based on the maximization of the following log-likelihood (LLK) function under normality given by the equation

$$\begin{aligned} \ell(Y_t; \mu, A, B, \Sigma_r, \Sigma_c) &= -m(T-1) \ln |\Sigma_c| - n(T-1) \ln |\Sigma_r| + \\ &\quad - \sum_{t=2}^T \text{tr} [\Sigma_r^{-1} (Y_t - \mu - AY_{t-1}B') \Sigma_c^{-1} (Y_t - \mu - AY_{t-1}B')] , \end{aligned}$$

where  $|\cdot|$  and  $\text{tr}(\cdot)$  denote the determinant and the trace of a matrix, respectively. It can be noted that these two equations add the matrix  $\mu$  inside equations (4) and (11) in the COMPSTAT.

Since for both estimation methods the MAR package adopts the so-called “flip-flop” iterative procedure [see, for example, Dutilleul, 1999], the inclusion of the intercept matrix modifies the definition of several matrices already defined in sections 2.3.1 and 2.3.2 of the COMPSTAT. In particular, after initializing the parameter matrices  $B_0$  as a full-rank random matrix of uniformly distributed numbers and  $\mu_0$  as the matrix of sample means of the data, the parameter matrices computed at each step  $\kappa \in [1, T-1]$  of the algorithm become respectively

$$\begin{cases} A_\kappa = \|[(Y_{t-1} - \mu_{\kappa-1})B_{\kappa-1}Y'_{t-1}](Y_{t-1}B'_{\kappa-1}B_{\kappa-1}Y'_{t-1})^{-1}\|_F \\ B_\kappa = [(Y_{t-1} - \mu_{\kappa-1})'A_\kappa Y_{t-1}](Y'_{t-1}A'_\kappa A_\kappa Y'_{t-1})^{-1} \\ \mu_\kappa = \frac{1}{T-1} [Y_t - A_\kappa Y_{t-1}B'_\kappa] \end{cases} \quad [\text{LSE estimator}]$$

and

$$\begin{cases} A_\kappa = \|[(Y_{t-1} - \mu_{\kappa-1}\Sigma_{c,\kappa-1}^{-1}B_{\kappa-1}Y'_{t-1}](Y_{t-1}B'_{\kappa-1}\Sigma_{c,\kappa-1}^{-1}B_{\kappa-1}Y'_{t-1})^{-1}\|_F \\ \Sigma_{r,\kappa} = \frac{1}{n(T-p)} \sum_{t=p+1}^T (Y_t - \mu_{\kappa-1} - A_\kappa Y_{t-1}B'_{\kappa-1})(Y'_t - \mu'_{\kappa-1} - B_{\kappa-1}Y'_{t-1}A'_\kappa) \\ B_\kappa = [(Y_{t-1} - \mu_{\kappa-1})'\Sigma_{r,\kappa}^{-1}A_\kappa Y_{t-1}](Y'_{t-1}A'_\kappa \Sigma_{r,\kappa}^{-1}A_\kappa Y'_{t-1})^{-1} \\ \mu_\kappa = \frac{1}{T-1} [Y_t - A_\kappa Y_{t-1}B'_\kappa] \\ \Sigma_{c,\kappa} = \frac{1}{m(T-p)} \sum_{t=p+1}^T (Y'_t - \mu'_\kappa - B_\kappa Y'_{t-1}A'_\kappa)(Y_t - \mu_\kappa - A_\kappa Y_{t-1}B'_\kappa) \end{cases} \quad [\text{MLE estimator}]$$

Note that, in the case of MLE estimation, the matrices  $A$  and  $\Sigma_c$  must also be initialized. In the former case, the initialization is  $A_0 = m^{-1/2}I_m$ , while in the latter case, the initial matrix  $\Sigma_{c,0}$  is defined according to the equation (3). The  $\Sigma_r$  matrix does not need to be initialized.

Once the matrices of the estimated parameters  $\hat{\mu}$ ,  $\hat{A}$  and  $\hat{B}$  are obtained from the above algorithm, the residual matrix is calculated by updating the equation (5) in the COMPSTAT according the formula

$$\hat{E}_t = Y_{t-1} - \hat{\mu} - \hat{A}Y_{t-1}\hat{B}'.$$

Finally, the stacked Jacobian matrix introduced in equation (8) updates as follows

$$\begin{aligned} W_t &= \begin{bmatrix} \frac{\partial \text{vec}(Y_t)}{\partial \mu} & \frac{\partial \text{vec}(Y_t)}{\partial A} & \frac{\partial \text{vec}(Y_t)}{\partial B} \end{bmatrix} \\ &= [I_{mn} \quad (BY'_{t-1}) \otimes I_m \quad I_n \otimes (AY_{t-1})']', \end{aligned} \quad (4)$$

where  $\mathbf{W}_t$  has dimension  $(m^2 + n^2 + mn) \times mn$ , while the vector  $\boldsymbol{\gamma}$  in equation (9) becomes

$$\boldsymbol{\gamma} = \begin{bmatrix} \mathbf{0}_{mn \times 1} & \text{vec}(\mathbf{A})_{m^2 \times 1} & \mathbf{0}_{n^2 \times 1} \end{bmatrix}'.$$

### 1.3 Package functions

The **MAR** package specifically delivers three public or primary functions to estimate the parameter matrices of a MAR(1) model, the related impulse response functions, and some residual tests. The public functions, which provide access to the Graphical User Interface (GUI) environment, are defined as follows.

#### Public functions

```
function scalar MAR (lists In, scalar lag, int intercept, int est_method,
                    scalar maxiter, bundle *bdl)
```

This function performs the MAR(1) estimation and creates the bundle, generating the necessary output upon which the impulse response and residual test functions depend. The arguments correspond to all the entries that the user must provide:

- **lists In**: a **lists** environment containing all the time series of interest, arranged as an array of lists (see **SCRIPT 1**);
  - **scalar lag**: scalar indicating the number of MAR lags. In the current version of the package, this value is fixed at 1. This value may be increased in future releases, keeping  $p = 1$  as the default value;
  - **int intercept demean**: this is the main difference in the actual version 1.0 of the package. The boolean switch **demean** is replaced by the ‘int’ **intercept** that allows the user to estimate the MAR(1) model in three different ways. The default value 1 is assigned to the model with constant, while the values 2 and 3 are given to models without any deterministic component. When the value 3 is set, the model is estimated on demeaned time series;
  - **int est\_method**: estimation method. As already stated, the **MAR** package allows the use of the LSE and the MLE estimators, which are assigned the numbers 1 and 2, respectively. The default estimator is LSE;
  - **scalar maxiter**: maximum number of iterations allowed in the MAR(1) estimation phase. The default is 100, but the user can choose any integer value. In some circumstances, choosing the number of iterations can be useful to prevent the numerical algorithm from continuing to iterate in the event of a failure to converge, or to be able to observe the estimated values of the parameters after a preset number of iterations;
  - **bundle bdl**: preset container for various objects created by the package. These objects can be used in the various functions in the package or retrieved after estimation. In particular, **MAR** package stores the following items into the bundle (the names of each object are in parentheses):
- scalars** — the number of variables in row  $m$  (**m**), the number of variables in column  $n$  (**n**), the sample size  $T$  (**T**), the number of lags  $p$  (**p**), the log-likelihood (**LLK**), the three information criteria (**AIC**, **BIC** and **HQC**), the number of iterations of the

algorithm and the maximum number of iterations set by the user (`iterations` and `maxiter`), the product  $\rho(\hat{\mathbf{A}})\rho(\hat{\mathbf{B}})$  (`check`);

**matrices** — vector containing all the standard errors (`SE`), the estimated parameter covariance matrix (`VCV`),  $\hat{\Sigma}$  (`Sigma`),  $\hat{\Sigma}_{ML}$  (`SigmaKron`),  $\hat{\Sigma}_r$  (`SigmaR`),  $\hat{\Sigma}_c$  (`SigmaC`), the initial matrix  $\mathbf{B}_0$  (`InitSigma`);

**array of matrices** — the estimated  $\mathbf{A}$ ,  $\mathbf{B}$  and eventually  $\boldsymbol{\mu}$  parameter matrices (`Theta`, length 3), the entire dataset whose each element corresponds to the data matrix  $\mathbf{Y}_t$  (`AllDataMat`, length  $T$ ), the fitted data (`Fitted`, length  $T$ ), and the model residuals (`Resids`, length  $T$ );

**list** — list containing all the time series of interest (`VarList`). The order follows that of  $\text{vec}(\mathbf{Y}_t)$ ;

**strings** — the estimation method (`method`), the names of the variables in the data set (`Vnames`), the sample start date (`startdate`) and the end date (`enddate`).

When the MAR estimation converges after fewer iterations than the maximum set by the user, the public function returns the scalar 1 (success); otherwise, it returns 0.

Once the MAR(1) estimation is performed, the user can call the following public functions.

```
function matrix MAR_IRFs (scalar s_i, scalar s_j, scalar horizon, int verb,
                          bundle *x)
```

When called after the MAR estimation, this function calculates the ‘generalized innovation impulse response functions’ [GIRFs, see Koop et al., 1996] in all the  $m \times n$  dependent variables after a shock in the  $(i, j)$ -th variable.<sup>1</sup> As in the public function, the following arguments must be supplied by the user

- **scalar s\_i**: the row index of the variable that has been shocked ( $s_i = 1, 2, \dots, m$ );
- **scalar s\_j**: the column index of the variable that has been shocked ( $s_j = 1, 2, \dots, n$ );
- **scalar horizon**: an integer positive scalar  $H$  indicating the number of periods on which the GIRFs have to be computed. The default value is 10 periods;
- **int verb**: this argument provides a print command. Specifically, there are three options available, namely “NONE”, “TABLE”, and the default “TABLE & PLOT”. By choosing the first option, the function calculates the GIRFs, but does not provide the values or graphs. By selecting the second option, a table with the GIRFs arranged by column is printed. By choosing the third option, the above table is printed, and a separate window with the plots appears;
- the bundle.

The function returns a matrix with  $H + 1$  rows and  $m \times n$  columns. Each row contains all the GIRFs calculated in each period  $h \in [0, H]$ , while the columns refer to the reactions of each variable following a shock on the  $(i, j)$ -th variable. The variables in the columns are arranged in the order given in  $\text{vec}(\mathbf{Y}_t)$ .

Note that if the first two arguments are out of bounds, a warning message appears, and the function does not calculate the GIRFs, returning the value 0 (failure).

---

<sup>1</sup>The GIRFs address the well-known problem of variable ordering [Pesaran and Shin, 1998]. Note that Chen et al. [2021] call the GIRFs obtained in this way ‘shock-first impulse response function with orthogonal innovations’ (s1-oIRF).

```
function matrix MAR_Diagnostics(int test, scalar lag, bool multivariate, bundle *x)
```

This function performs standard diagnostic tests on the model's residuals. It takes four arguments.:

- **int test**: this argument indicates the test type. The user can choose between 0 (All diagnostics, the default value), 1 (Autocorrelation), 2 (Heteroskedasticity), and 3 (Normality). Selecting the default option runs and reports all diagnostic tests;
- **scalar lag**: the lag for autocorrelation and heteroskedasticity tests. The default value is 1 lag;
- **bool multivariate**: a Boolean switch that refers to the choice of running multivariate diagnostic tests or a battery of univariate diagnostic tests. All tests are run on the residual time series  $\hat{E}_t$ , exploiting `gretl`'s `modtest` and `normtest` functions. In the multivariate setting, the Rao F test for joint autocorrelation, the joint ARCH-LM test for conditional heteroskedasticity, and the Doornik-Hansen test for joint normality are conducted. In the univariate setting, the tests are Ljung-Box, ARCH, and Jarque-Bera;
- the bundle.

Since diagnostic tests can be called together or separately, this function returns a matrix where each row refers to the test performed, while the columns contain the test statistics and the corresponding  $p$ -values in its columns. When the multivariate setting is selected, an additional column with the number of lags is added as the first column (this column contains NA for the joint normality test).

## Private functions

The choice regarding the value of the above mentioned **intercept** is assigned to variable in the bundle, therefore there are not relevant changes in the private functions. Clearly, the eventual presence of a constant in the model is now considered inside each function.

There are only a couple of exceptions.

```
function matrices PrepareData (lists In, matrix *SM)
```

This function has been slightly modified from the previous version. The new feature is the calculation of sample means of the data, which are arranged in a matrix of size  $m \times n$ . The address `*SM` has been added to obtain this matrix. The sample mean matrix can be extracted from the package using the `SampleMeans` ticker.

```
function matrices DeMean (matrices M, matrix Mean)
```

This is a new function that is applied to an array of matrices (**matrices**) and returns an array of matrices with the same dimensions. Specifically, it transforms each matrix containing data in a matrix that contains the deviations of such data from their sample time averages contained in matrix `Mean`.

## 1.4 The GUI

Compared to the GUIs presented in the COMPSTAT, only the main one shows a slight change, since the “Demean time series” boolean switch is replaced by the “Model” dialog box, where

the default value is “with constant”. The other available options are “without constant” and “without constant (demeaned time series)” in order to preserve the two alternatives from the previous version of the package. Everything else remained unchanged.

Figure 1: The GUIs  
(a) MAR public function

(b) MAR\_IRFs public function

(c) MAR\_Diagnostics public function

**Note:** The MAR function must be called first, since the other two functions require its output to be executed.

Once installed, the package offers the option of attaching a Graphical User Interface (hereafter GUI) to the main `gretl` window via the `Model/Multivariate time series/Matrix AR(1)` path. This is an excellent way to access the package’s functionality, but it should be noted that the full degree of flexibility and customization can only be achieved via scripting. A dialog box appears asking the user to select one of three alternative functions ‘MAR’, ‘IRF’ or ‘Diagnostics’. Selecting a function opens one of the dialog boxes shown in Figure 1, where the values entered are the default values of the different public functions. Clearly, all the fields correspond to the arguments of these functions, although they are presented in a simplified manner for ease of use.

## 2 Example file

The proposed example substantially follows the one in the COMPSTAT, since it uses the same data, the same arrangement of the time series in an array of lists and calls the same diagnostics and impulse response functions (hereafter, IRFs). In particular, SCRIPT 1 below introduces the scalar `intercept` which assumes the default value 1, so that `MAR(1)` contains the constant.

SCRIPT 1

```
set verbose off
include MAR.gfn

#Opening the file containing data
open Datavec.gdt --frompkg=MAR

#Setting a seed for random number generation
set seed 15061973

#Choosing the maximum number of 'flip-flop' iterations
scalar maxiter = 100

#Number of lags
scalar p=1

#constant matrix in the MAR(1)
scalar intercept=1

#Defining the bundle
bundle bdl

# Preparation of n lists (in columns) containing m variables (in rows)
list S1=V1 V2
list S2=V3 V4
list S3=V5 V6
list S4=V7 V8

#Creating the array of lists required by the public function
lists X=defarray(S1,S2,S3,S4)

# LSE estimation
MAR(X,p,intercept,1,maxiter,&bdl)

# MLE estimation
MAR(X,p,intercept,2,maxiter,&bdl)

# Multivariate diagnostics
MAR_Diagnostics(,4,1,&bdl)
# Multivariate diagnostics
MAR_Diagnostics(,4,0,&bdl)

# GIRFs
MAR_IRFs(1,2,10,,&bdl)

print "-----"

# Print the contents of the bundle
bdl
```

By running this example file, the LSE and MLE estimates contained in Tables 1 and 2 are produced. In this context, the diagnostics and the IRFs are calculated by using these results,

but they are obtained with the same commands of the COMPSTAT, therefore they are omitted for brevity.

Table 1: LSE estimation with constant

MAR(1), using observations 2021-01-01:2024-12-24 (T = 1038).

Each data matrix at time t has dimension 2x4.

Estimation method: LSE.

Convergence achieved after 22 iterations.

Intercepts:

	coefficient	std. error	z	p-value	
-----					
const[1,1]	4.39892	0.211696	20.78	6.65e-96	***
const[2,1]	0.0636315	0.0518498	1.227	0.2197	
const[1,2]	0.988090	0.234022	4.222	2.42e-05	***
const[2,2]	0.0226785	0.0737919	0.3073	0.7586	
const[1,3]	2.36532	0.227912	10.38	3.12e-25	***
const[2,3]	0.0697173	0.0618376	1.127	0.2596	
const[1,4]	0.590291	0.243462	2.425	0.0153	**
const[2,4]	0.0511778	0.0597859	0.8560	0.3920	

Matrix A:

	coefficient	std. error	z	p-value	
-----					
A[1,1]	0.985857	0.0273460	36.05	1.32e-284	***
A[2,1]	0.00346225	0.0172827	0.2003	0.8412	
A[1,2]	-0.165729	0.162561	-1.019	0.3080	
A[2,2]	0.0246406	0.0587583	0.4194	0.6750	

Matrix B:

	coefficient	std. error	z	p-value	
-----					
B[1,1]	-0.250138	0.0320943	-7.794	6.50e-15	***
B[2,1]	0.156746	0.0349833	4.481	7.44e-06	***
B[3,1]	-0.0359331	0.0338396	-1.062	0.2883	
B[4,1]	0.0517511	0.0361449	1.432	0.1522	
B[1,2]	0.169890	0.0322632	5.266	1.40e-07	***
B[2,2]	0.0545633	0.0353518	1.543	0.1227	
B[3,2]	0.171187	0.0346535	4.940	7.81e-07	***
B[4,2]	0.165614	0.0370045	4.476	7.62e-06	***
B[1,3]	0.00139078	0.0327058	0.04252	0.9661	
B[2,3]	0.115157	0.0361486	3.186	0.0014	***
B[3,3]	-0.139492	0.0354641	-3.933	8.38e-05	***
B[4,3]	0.111801	0.0375868	2.974	0.0029	***
B[1,4]	0.0851503	0.0304594	2.796	0.0052	***
B[2,4]	0.199273	0.0340493	5.852	4.84e-09	***
B[3,4]	0.232876	0.0333734	6.978	3.00e-12	***
B[4,4]	0.301679	0.0359725	8.386	5.01e-17	***

Log-likelihood -25995.73 Akaike criterion 52047.46

Schwarz criterion 52185.92 Hannan-Quinn 52099.99

Check for time series stationarity:

$\rho(A)\rho(B)=0.48471$

The estimated MAR(1) model is stationary and causal.



Table 2: MLE estimation with constant

MAR(1), using observations 2021-01-01:2024-12-24 (T = 1038).  
Each data matrix at time t has dimension 2x4.  
Estimation method: MLE.  
Convergence achieved after 20 iterations.

Intercepts:

	coefficient	std. error	z	p-value	
const[1,1]	4.39640	0.227785	19.30	5.31e-83	***
const[2,1]	0.0623313	0.0486109	1.282	0.1998	
const[1,2]	1.01175	0.280126	3.612	0.0003	***
const[2,2]	0.0273283	0.0629371	0.4342	0.6641	
const[1,3]	2.34779	0.258619	9.078	1.10e-19	***
const[2,3]	0.0701157	0.0556209	1.261	0.2075	
const[1,4]	0.596933	0.240754	2.479	0.0132	**
const[2,4]	0.0574473	0.0559472	1.027	0.3045	

Matrix A:

	coefficient	std. error	z	p-value	
A[1,1]	0.994641	0.0150118	66.26	0.0000	***
A[2,1]	0.00209584	0.0123701	0.1694	0.8655	
A[1,2]	-0.0961620	0.154664	-0.6217	0.5341	
A[2,2]	-0.0379111	0.0420839	-0.9008	0.3677	

Matrix B:

	coefficient	std. error	z	p-value	
B[1,1]	-0.247771	0.0336744	-7.358	1.87e-13	***
B[2,1]	0.155656	0.0412653	3.772	0.0002	***
B[3,1]	-0.0345030	0.0380328	-0.9072	0.3643	
B[4,1]	0.0509110	0.0354080	1.438	0.1505	
B[1,2]	0.166991	0.0341708	4.887	1.02e-06	***
B[2,2]	0.0516116	0.0419252	1.231	0.2183	
B[3,2]	0.166709	0.0387629	4.301	1.70e-05	***
B[4,2]	0.162010	0.0361125	4.486	7.25e-06	***
B[1,3]	0.00213111	0.0348900	0.06108	0.9513	
B[2,3]	0.114661	0.0428917	2.673	0.0075	***
B[3,3]	-0.135243	0.0396881	-3.408	0.0007	***
B[4,3]	0.112611	0.0368538	3.056	0.0022	***
B[1,4]	0.0831070	0.0323852	2.566	0.0103	**
B[2,4]	0.195866	0.0399185	4.907	9.27e-07	***
B[3,4]	0.230265	0.0369321	6.235	4.52e-10	***
B[4,4]	0.299333	0.0345326	8.668	4.39e-18	***

Log-likelihood      -25992.2   Akaike criterion      52040.4  
Schwarz criterion   52178.86   Hannan-Quinn      52092.93

Check for time series stationarity:

$$\rho(A)\rho(B)=0.48336$$

The estimated MAR(1) model is stationary and causal.

It is worth noting that the estimates obtained on the deviations of the data from their time averages given in Tables 1 and 2 of the COMPSTAT can be performed by setting the scalar `intercept` to the value 3; if instead `intercept= 2` is set, the LSE and/or MLE estimates of a MAR(1) without constant estimated directly on the available data can be obtained (this exercise is left to the user).

### 3 Change log

#### version 0.1 (June 2025)

This is the original version of the `MAR` package, which has never been loaded into `gretl`. This is the version previously presented at the 9th biennial `gretl` Conference, held in Birmingham, UK, on June 19-20, 2025, and subsequently published in the paper by Bucci, Palomba, and Tedeschi [2026].

#### version 1.0 (May 2026)

This is the first release in which all public and private functions have been updated to take into account the intercept matrix in the MAR(1) model. Additionally, the public functions `IFRs` and `Diagnostics` have been renamed `MAR_IRFs` and `MAR_Diagnostics` respectively. The estimation output is now produced using the `gretl`'s dedicated function `modprint`.

### References

- Andrea Bucci, Giulio Palomba, and Marco Tedeschi. Matrix-valued autoregressive (MAR) models in `gretl`. *Computational Statistics*, 41, 2026. URL <https://doi.org/10.1007/s00180-025-01681-8>. article 47.
- Rong Chen, Han Xiao, and Dan Yang. Autoregressive models for matrix-valued time series. *Journal of Econometrics*, 222(1):539–560, 2021.
- Pierre Dutilleul. The MLE algorithm for the matrix normal distribution. *Journal of Statistical Computation and Simulation*, 64(2):105–123, 1999.
- Gary Koop, M Hashem Pesaran, and Simon M Potter. Impulse response analysis in nonlinear multivariate models. *Journal of econometrics*, 74(1):119–147, 1996.
- M.Hashem Pesaran and Yongcheol Shin. Generalized impulse response analysis in linear multivariate models. *Economics Letters*, 58(1):17–29, 1998. ISSN 0165-1765.