Building Better Forecasting Models with Transfer Functions

Jian Cao
JMP Division, SAS Institute

Discovery Summit Europe
Copenhagen 03.17.2019

Abstract

How to model and forecast the time series when it is interrupted due to interventions (e.g., process changes)? If you have leading indicators or other exogenous variables how can you incorporate them into your ARIMA models to make better forecast?

In this paper I will try to demystify the transfer function models in JMP with key use cases: Regression with ARIMA Errors, Distributed Lag Models and Intervention Models. I will demonstrate the benefits of using the transfer functions over OLS regression and ARIMA for building better forecasting models.

1.Introduction

In this paper I discuss statistical models for forecasting with time series data. My primary focus is on using the transfer functions to build better models for forecasting. In their popular 2018 book, *Forecasting: Principles and Practices*[1], Rob J. Hyndman and George Athanasopoulos discuss several modeling approaches: OLS (Ordinary Least Squares) regression models, ARIMA (Auto-Regressive Integrated Moving Average) models and dynamic models (i.e., transfer function models). Their book is free on line, so I will adopt one of their examples to build the different types of models to demonstrate why the transfer function model makes better forecast.

In Section 3, following an introduction to the general Transfer Function Model I'll discuss the key use cases with examples: Regression with ARIMA Errors, Distributed Lag Models, and Intervention Models. A summary is provided in Section 4.

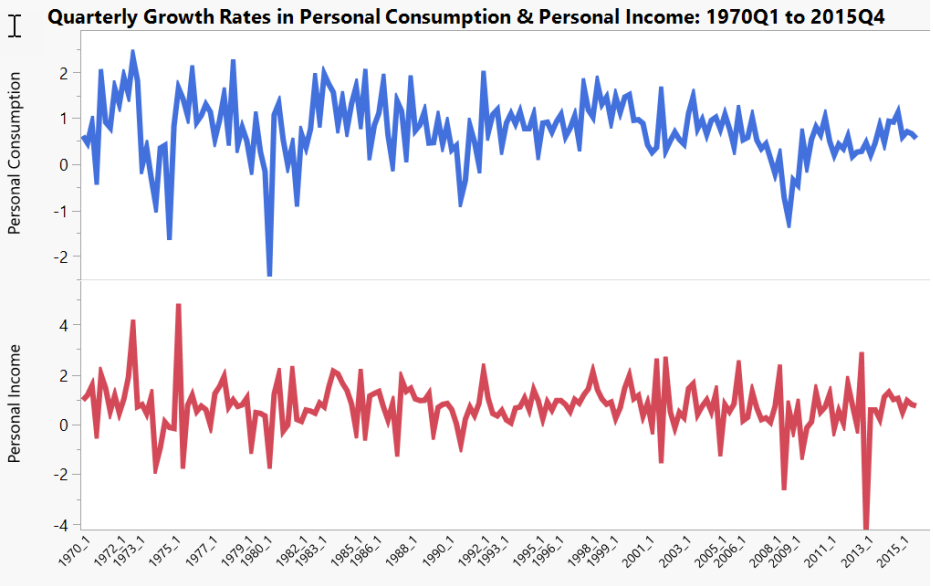
2. Why Transfer Function Models Are Better: An Example

In this section I build and compare three different types of forecasting models. The data are quarterly growth rates (i.e., quarter-over-quarter percentage changes) of personal consumption

[1] The book is free at *https://otexts.com/fpp2/forecasting.html*

expenditure and personal disposable income, both in real dollars, for the US from 1970Q1 to 2015Q4 (Figure 1). The objective is predicting the *Personal Consumption Growth* for the future quarters. *Personal Disposal Income* is considered a prediction variable.

Figure 1: Quarterly Growth Rates of Personal Consumption and Personal Disposable Income in the US: 1970Q1 to 2015Q4



2.1 OLS Regression

When it comes to statistical modeling the OLS regression is perhaps the most popular method. For this data I fit the following regression model by OLS (Figure 2):

$$\hat{y}_t = 0.55 + 0.28 * x_t \qquad \text{Eq. 1}$$

x_t and y_t are the Personal Income Growth Rate and Personal Consumption Growth Rate at time t. Since the Income Growth is available for 2016Q1 I will use it to predict the Consumption Growth and compare the forecast to the actual consumption growth rate. See more on this in Section 2.4.

Figure 2: OLS Regression Fit Details

Summary of Fit

RSquare	0.159025
RSquare Adj	0.154479
Root Mean Square Error	0.602608
Mean of Response	0.746471
Observations (or Sum Wgts)	187

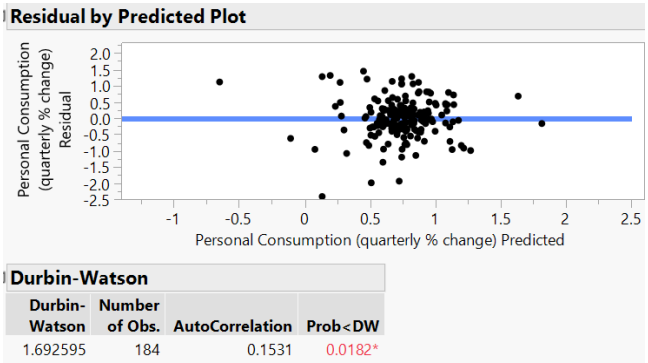
Parameter Estimates

Term	Estimate	Std Error	t Ratio	Prob>|t|
Intercept	0.5451038	0.055687	9.79	<.0001*
Personal Income (quarterly % change)	0.2806012	0.047442	5.91	<.0001*

A key assumption under OLS regression is the independence of regression errors. Clearly, it is violated as suggested by the **Durbin-Watson** test (0,153, *p-value* =0.018) in Figure 3. That is, OLS residuals are not independent (see **Residuals by Predictions Plot** in Figure 3**).** The

presence of autocorrelations in time series data is not surprising because the value of a variable in the current time period is likely correlated with its past values. However, autocorrelated errors cause the OLS standard errors and test statistics to be invalid, thus rendering the prediction misleading.

Figure 3: Residuals Plot and Durbin-Watson Test



2.2 ARIMA Models

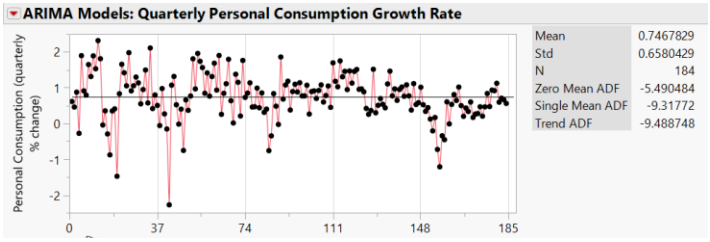
To explicitly consider the temporal correlations in a time series we can specify the output series $\boldsymbol{y_t}$ as a linear function of its past values:

$$\boldsymbol{y_t = \mu + \varphi_1 y_{t-1} + \varphi_1 y_{t-1} + \varphi_2 y_{t-2} + \cdots + \varphi_p y_{t-p} + e_t} \qquad \text{Eq. 2}$$

This is known as an auto-regressive (AR) model of order *p*. A more general form of this class of timeseries models is developed by Box and Jenkins, referred to as ARIMA (Auto-Regressive Integrated Moving Average) models[2]. They have outlined the steps to build an ARIMA model: data transformation, identification, estimation, diagnostic checking and forecasting. Note that a pure AR (or ARIMA) model does not include any input series as predictors.

ARIMA models are available in the *Time Series Platform* under JMP's ***Analyze*** => ***Specialized Modeling*** menu. In our example, *Personal Consumption* is already expressed in quarterly percent change, so no additional differencing is needed. This is confirmed by the **Trend ADF** (Augmented Dickey-Fuller) test in Figure 4. Also notice that no seasonality is present in the data (Figure 4).

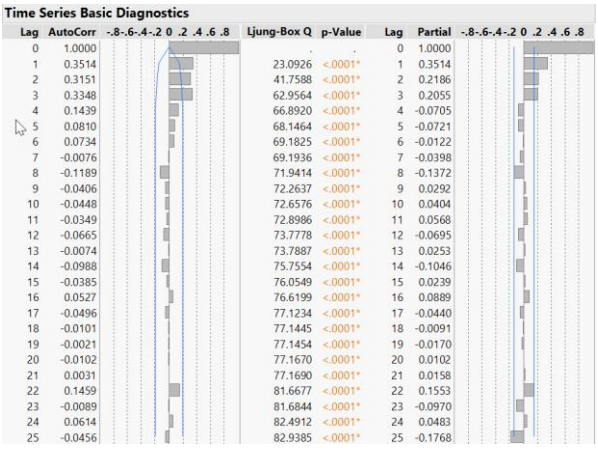
Figure 4: Time Series Plot and ADF Tests



[2] George Box, Gwilym Jenkins, Gregory Reinsel and Greta Ljung (2016). *Time Series: Forecasting and Control*.

Following the Box-Jenkins ARIMA methodology I check the ACF (auto-correlation function) and PACF (partial auto-correlation function) plots (Figure 5) and then run a set of non-seasonal ARIMA models: AR order ranges from 0 to 3, MA order ranges from 0 to 3 and I=0 (i.e., no differencing), as shown in Figure 6:

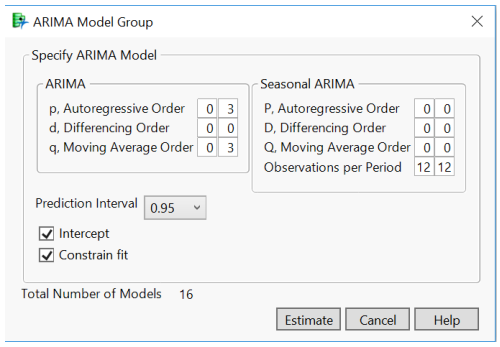
Figure 5: Time Series Basic Diagnostics Plots



Time Series Basic Diagnostics

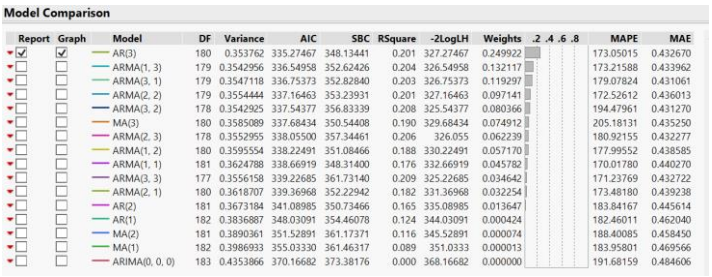
Lag	AutoCorr	-.8 -.6 -.4 -.2 0 .2 .4 .6 .8	Ljung-Box Q	p-Value	Lag	Partial	-.8 -.6 -.4 -.2 0 .2 .4 .6 .8
0	1.0000		.	.	0	1.0000	
1	0.3514		23.0926	<.0001*	1	0.3514	
2	0.3151		41.7588	<.0001*	2	0.2186	
3	0.3348		62.9564	<.0001*	3	0.2055	
4	0.1439		66.8920	<.0001*	4	-0.0705	
5	0.0810		68.1464	<.0001*	5	-0.0721	
6	0.0734		69.1825	<.0001*	6	-0.0122	
7	-0.0076		69.1936	<.0001*	7	-0.0398	
8	-0.1189		71.9414	<.0001*	8	-0.1372	
9	-0.0406		72.2637	<.0001*	9	0.0292	
10	-0.0448		72.6576	<.0001*	10	0.0404	
11	-0.0349		72.8986	<.0001*	11	0.0568	
12	-0.0665		73.7778	<.0001*	12	-0.0695	
13	-0.0074		73.7887	<.0001*	13	0.0253	
14	-0.0988		75.7554	<.0001*	14	-0.1046	
15	-0.0385		76.0549	<.0001*	15	0.0239	
16	0.0527		76.6199	<.0001*	16	0.0889	
17	-0.0496		77.1234	<.0001*	17	-0.0440	
18	-0.0101		77.1445	<.0001*	18	-0.0091	
19	-0.0021		77.1454	<.0001*	19	-0.0170	
20	-0.0102		77.1670	<.0001*	20	0.0102	
21	0.0031		77.1690	<.0001*	21	0.0158	
22	0.1459		81.6677	<.0001*	22	0.1553	
23	-0.0089		81.6844	<.0001*	23	-0.0970	
24	0.0614		82.4912	<.0001*	24	0.0483	
25	-0.0456		82.9385	<.0001*	25	-0.1768	

Figure 6: Specify ARIMA Model Orders



A total number of 16 models are fit. The **Model Comparison** ranks AR(3), i.e., ARIMA (3,0,0), as the best model as it has the smallest AIC value (Figure 7). The fit details for the selected model are shown in Figure 8.

Figure 7: Goodness-of-Fit Statistics for Each Model



Model Comparison

Report	Graph	Model	DF	Variance	AIC	SBC	RSquare	-2LogLH	Weights	.2 .4 .6 .8	MAPE	MAE
☑	☑	AR(3)	180	0.353762	335.27467	348.13441	0.201	327.27467	0.249922		173.05015	0.432670
☐	☐	ARMA(1, 3)	179	0.3542956	336.54958	352.62426	0.204	326.54958	0.132117		173.21588	0.433962
☐	☐	ARMA(3, 1)	179	0.3547118	336.75373	352.82840	0.203	326.75373	0.119297		179.07824	0.431061
☐	☐	ARMA(2, 2)	179	0.3554444	337.16463	353.23931	0.201	327.16463	0.097141		172.52612	0.436013
☐	☐	ARMA(3, 2)	178	0.3542925	337.54377	356.83339	0.208	325.54377	0.080366		194.47961	0.431270
☐	☐	MA(3)	180	0.3585089	337.68434	350.54408	0.190	329.68434	0.074912		205.18131	0.435250
☐	☐	ARMA(2, 3)	178	0.3552955	338.05500	357.34461	0.206	326.055	0.062239		180.92155	0.432277
☐	☐	ARMA(1, 2)	180	0.3595554	338.22491	351.08466	0.188	330.22491	0.057170		177.99552	0.438585
☐	☐	ARMA(1, 1)	181	0.3624788	338.66919	348.31400	0.176	332.66919	0.045782		170.01780	0.440270
☐	☐	ARMA(3, 3)	177	0.3556158	339.22685	361.73140	0.209	325.22685	0.034642		171.23769	0.432722
☐	☐	ARMA(2, 1)	180	0.3618707	339.36968	352.22942	0.182	331.36968	0.032254		173.48180	0.439238
☐	☐	AR(2)	181	0.3673184	341.08985	350.73466	0.165	335.08985	0.013647		183.84167	0.445614
☐	☐	AR(1)	182	0.3836887	348.03091	354.46078	0.124	344.03091	0.000424		182.46011	0.462040
☐	☐	MA(2)	181	0.3890361	351.52891	361.17371	0.116	345.52891	0.000074		188.40085	0.458450
☐	☐	MA(1)	182	0.3986933	355.03330	361.46317	0.089	351.0333	0.000013		183.95801	0.469566
☐	☐	ARIMA(0, 0, 0)	183	0.4353866	370.16682	373.38176	0.000	368.16682	0.000000		191.68159	0.484606

Figure 8: Fit Details for the Best ARIMA Model

Model: ARIMA(3,0,0)

Model Summary

DF	180	Stable	Yes
Sum of Squared Errors	63.6771524	Invertible	Yes
Variance Estimate	0.35376196		
Standard Deviation	0.59477891		
Akaike's 'A' Information Criterion	335.274666		
Schwarz's Bayesian Criterion	348.134409		
RSquare	0.20066642		
RSquare Adj	0.1873442		
MAPE	173.050153		
MAE	0.43267005		
-2LogLikelihood	327.274666		

Parameter Estimates

Term	Lag	Estimate	Std Error	t Ratio	Prob>|t|	Constant Estimate	Mu
AR1	1	0.22846781	0.0718675	3.18	0.0017*	0.30326402	0.74287975
AR2	2	0.16081575	0.0727857	2.21	0.0284*		
AR3	3	0.20248884	0.0716169	2.83	0.0052*		
Intercept	0	0.74287975	0.1046468	7.10	<.0001*		

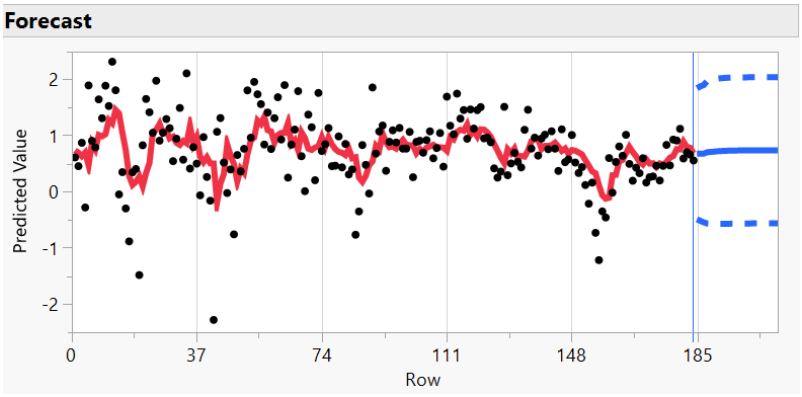
The ARIMA (3,0,0) is:

$$\widehat{y_{t+1}} = 0.30 + 0.23y_t + 0.16y_{t-1} + 0.20y_{t-2} \qquad \text{Eq. 3}$$

where y_t , y_{t-2} and y_{t-3} are the consumption growth rates at current and two previous quarters. This means that next quarter's consumption growth rate depends on past two quarters' growth.

In Figure 9, the red line shows how the above model fits the historical personal consumption growth pattern from 1970Q1 to 2015Q4, and solid blue line and dotted blue lines show the point forecasts and their 95% confidence limits, respectively. (Numerical forecast values can be saved to a new table.)

Figure 9: Forecasting Graph



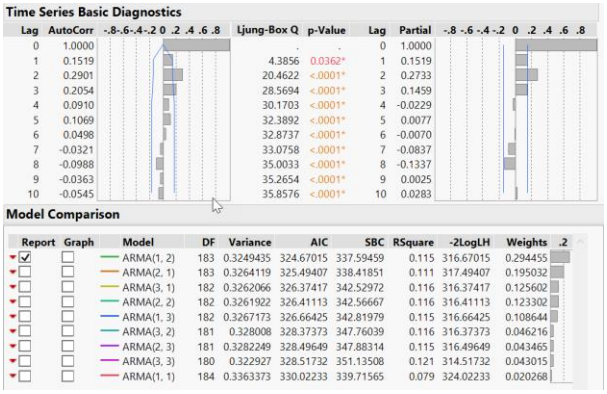
2.3 Regression with ARIMA Errors using Transfer Functions

The above ARIMA model does not explicitly include any input series such as income growth that may help predict consumption growth. On the other hand, as we have seen in Figure 3, the OLS regression violates a key assumption of statistical independence of the errors. We now turn to the Transfer Function Models that allow predictors and correct for autocorrelations.

A critical step in building a transfer function model is identification of the ARIMA structure for the autocorrelated errors. As with building the pure ARIMA models, the ACF and PACF on the saved OLS residuals from Eq. 1 are used to select possible orders of ARIMA (Figure 10). Then a group of ARIMA models are fit for the selection of the best structure.

Model Comparison report below suggests that an ARMA (1,2) (i.e., ARIMA (1,0,2)) is the best model (Figure 10).

Figure 10: Select the ARIMA for the regression errors



Now I use the **Transfer Function** option in the Time Series Platform to fit a regression model with *Personal Income* as an input series and an ARIMA (1,0,2) for noise series (Figure 11). The results are shown in Figure 12.

Figure 11: Specify a Regression with Non-seasonal ARIMA (1,0,2) Errors

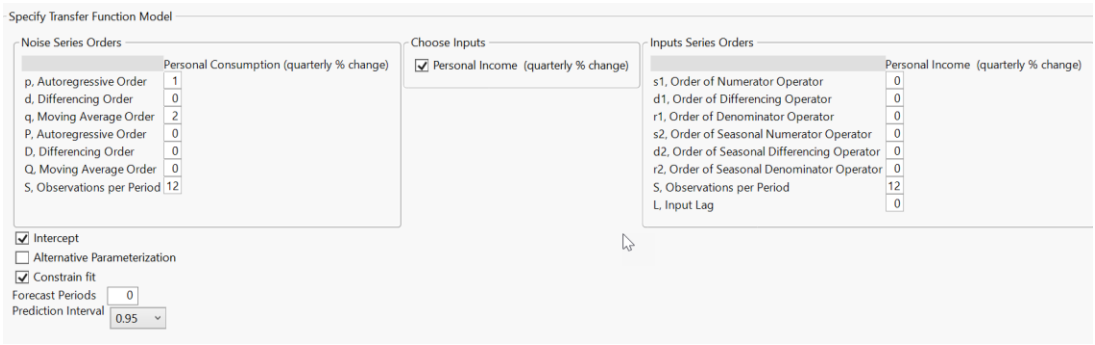
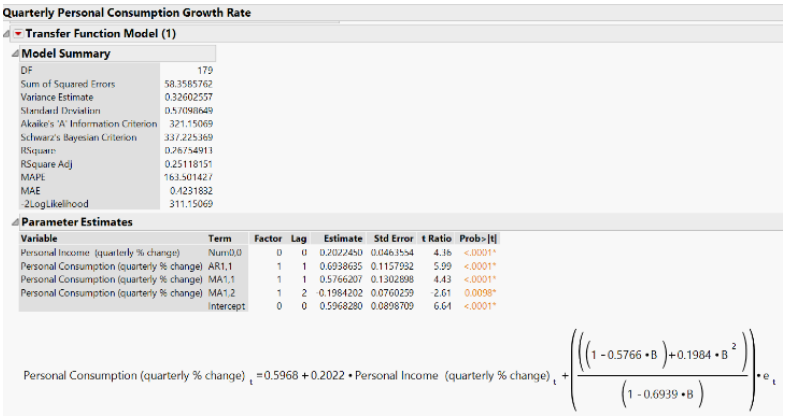


Figure 12: Regression with ARIMA (1,2) Errors



After checking ACF and PACF of the residuals I find no autocorrelation left and therefore will use the model to forecast. JMP will prompt you to provide the values for income growth rate to compute the consumption growth rates in the future.

Note that the above transfer function model may be alternatively written as the regression model with the noise term η_t:

$$\boldsymbol{y_t = 0.5968 + 0.2022 x_t + \eta_t} \qquad \text{Eq. 4}$$

Where η_t following an ARMA (1,2) process: $0.6939\eta_{t-1} + e_t - 0.5766e_{t-1} + 0.1984e_{t-2}$. ($\eta_{t-1}$ is the ARMA error at t-1, e_t, e_{t-1} and e_{t-2} are the random errors). And y_t and x_t are, respectively, the consumption growth rate and income growth rate.

Compared to the OLS model (Eq 1.), Eq 4. contains an ARMA error term to correct for the autocorrelations.

2.4 Does the Regression with ARIMA Errors forecast better than the OLS or ARIMA models?

$R^2_{adj.}$ for the three models, OLS, ARIMA (3,0,0) and the regression with ARIMA errors that we fit are 0.154, 0.187 and 0.251, respectively. I also generated the 'one-step-ahead" forecast using the actual personal income growth rate in 2016Q1 for the models: 0.691, 0.682, and 0.636. Compared to the actual consumption growth rate of 0.405, the regression with ARIMA errors model yields a smaller forecasting error. The confidence intervals are also narrower than those from the pure ARIMA model because we can explain some of the variation in consumption growth using the income predictor.

In summary, OLS regression models are static in the sense that the consumption growth is taken to be a linear function of income growth observed at the same point in time. On the other hand, pure ARIMA models the temporal dependence of the output time series but does not include the input series as predictors. Our better forecast comes from the model built with the transfer functions that combines both the time series dynamics and prediction variables.

3. Transfer Function Models and Use Cases

Regression with ARIMA Errors that we fit above falls within the so-called **Transfer Function Models**. There are other types of time series models that can be fit with the transfer functions.

Following an introduction to the general transfer function models in JMP, I'll discuss key use cases with examples.

3.1 Statistical Details[3]

The general form of the Transfer Function Model with a single input time series x_t (or a difference of x_t) at t. is:

$$(\mathbf{1}-\boldsymbol{B})^{\boldsymbol{d}}\boldsymbol{y_t} = \boldsymbol{\mu} + \frac{\boldsymbol{\omega(B)}}{\boldsymbol{\delta(B)}}\boldsymbol{x_{t-l}} + \frac{\boldsymbol{\theta(B)}}{\boldsymbol{\varphi(B)}}\boldsymbol{e_t} \qquad \text{Eq. 5}$$

The ratio of two polynomials, $\frac{\omega(B)}{\delta(B)}$, is the transfer function that specifies how the input series dynamically affects the output series, y_t. If there is a time delay (e.g., 2 periods delay) before any changes in x_t is felt in y_t, then use the input lag, l, to specify the delay (i.e., l=2). The error term, $\frac{\theta(B)}{\varphi(B)}e_t$, follows an ARIMA process representing the error dynamics of the model.

Additionally,

$\boldsymbol{\mu}$ -- Output series mean
$\boldsymbol{B}$ -- Backshift (i.e., lag) operator, e.g., $By_t = y_{t-1}$
$\boldsymbol{d}$ -- Degree of differencing. E.g., $(1-B)^2 y_t = ((y_t - y_{t-1}) - (y_{t-1} - y_{t-2}))$
$\boldsymbol{\omega(B)}$ -- Numerator polynomials for input series. For example, a polynomial of order 3 is $\omega_0 - \omega_1 B - \omega_2 B^2 - \omega_3 B^3$. Together with the step lag this specifies when and how long the input impact will last.
$\boldsymbol{\delta(B)}$ -- Denominator polynomials for input series. For example, a polynomial of order 1, $1-\delta B$, introduces exponentially weighted, infinite distributed lags into the transfer function as $\frac{\mathbf{1}}{1-\delta B} = 1 + \delta B + (\delta B)^2 + (\delta B)^3 + (\delta B)^4 + \cdots$.
$\boldsymbol{\varphi(B)}$ and $\boldsymbol{\theta(B)}$ **--** Auto-regressive part and moving average part of the model error term[4].
$\boldsymbol{e_t}$ -- White noise (i.e., random error).

The key difference between the pure ARIMA model and Transfer Function Model is the latter explicitly models the systematic dynamics of the input time series.

[3] For the ease of exposition, I only show the transfer function model with a single input series, but it can be generalized to multiple input series.
[4] For details on the AR and MA notations see *JMP® 14 Predictive and Specialized Modeling (*Pages 343-345). Or visit online at *https://www.jmp.com/support/help/14-2/statistical-details-for-the-time-series-platform.shtml#134469*.

Use Case 1: Regression with ARIMA Errors

This is the multiple regression with errors corrected for autocorrelations, as we discussed in Section 2.3.

For multiple input series, $x_{t1}, x_{t2} \ldots, x_{tk}$, the regression with ARIMA errors is:

$$\boldsymbol{y_t = \mu + \omega_1 x_{t1} + \omega_2 x_{t2} + \cdots + \omega_k x_{tk} + \eta_t} \quad \text{where the error } \boldsymbol{\eta_t = \frac{\theta(B)}{\varphi(B)} e_t} \qquad \text{Eq. 6}$$

Eq 6. is also known as Autoregressive Error Models.

Use Case 2: Distributed Lag Models

In the previous example we modeled the contemporaneous effects of income on consumption but did not consider any delayed income effects. The models that allow the impact of an input series to be distributed over past lags are the Distributed Lag Models.

One form of the Distributed Lag Model with a single input series is:

$$\boldsymbol{(1-B)^d y_t = \mu + \omega(B) x_{t-l} + \eta_t} \quad \text{where } \boldsymbol{\eta_t = \frac{\theta(B)}{\varphi(B)} e_t} \qquad \text{Eq. 7}$$

The polynomials for the numerator in the transfer function, $\omega(B)$, represent the dynamic effects of the input over different timer periods. For example, a polynomial of order 3, $\omega(B) = \omega_0 x_t - \omega_1 x_{t-1} - \omega_2 x_{t-2} - \omega_3 x_{t-3}$, specifies the dynamic impact of x_t up to three lags.

If the effects of x_t is exponentially decreasing a first-order of polynomials for the denominator, $\delta(B) = 1 - \delta_1 B$, may be used:

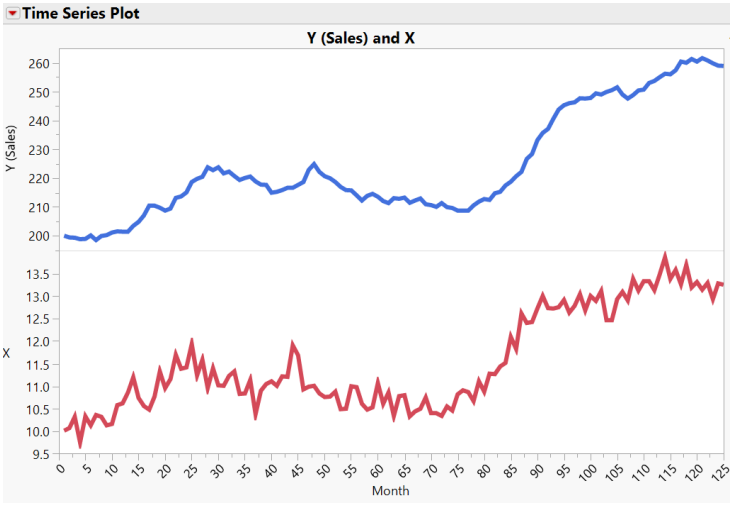
$$\boldsymbol{(1-B)^d y_t = \mu + \frac{\omega_0}{1-\delta_1 B} x_{t-l} + \eta_t} \qquad \text{Eq. 8}$$

An example of this form of the distributed lag model is illustrated below.

Example 2: Forecasting Sales using Lagged Predictors

The data used in this example, called *Series M*, comes from the classical book by George Box *et al*. (2016). We are interested in making forecasts for sales with a predictor, x (Figure 13). For model validation we will use the first 125 observations out of the 150 in the data set to build the models and the last 25 observations as the test set.

Figure 13: Monthly Sales and Input Series, X



A critical step in building a transfer function model is to identify *when* the input effect takes place, *how long* it lasts and in *what* shape it decays. This is done through **Prewhitening**. The results are the Prewhiten Corr, or the cross-correlations (Figure 14): it starts at lag 3 and then fades away gradually. This implies a transfer function of the form $\frac{\omega_0}{1-\delta B}$ for the shape of the dynamics. Specifically, the **Prewhitening Plot** in Figure 13 says that starting at lag 3, an input's impact declines exponentially according to this: $\omega_0(1-B)x_{t-3} + \omega_0\delta(1-B)x_{t-4} + \omega_0\delta^2(1-B)x_{t-5} + \omega_0\delta^3(1-B)x_{t-6} + \omega_0\delta^4(1-B)x_{t-7} + \cdots$.

Figure 14: Cross-correlations between the output and input

Prewhitening Plot (0,1,1)(0,0,0)12 : Output - Y (Sales) vs. Input - X

Lag	Prewhiten Corr
-2	0.0067
-1	0.1059
0	0.0907
1	0.1012
2	0.0538
3	0.6810
4	0.4666
5	0.3457
6	0.2813
7	0.2780
8	0.2271
9	0.1838
10	0.0968
11	0.1403
12	0.0748
13	0.0071
14	0.0954
15	0.0561
16	0.0308
17	-0.0149
18	-0.0607
19	-0.0244
20	0.0090
21	-0.0543
22	-0.0044
23	0.0775
24	0.0504
25	0.0338

Lag	Noise AutoCorr
23	-0.0055
24	-0.0542
25	0.0124

A preliminary distributed lag model (i.e., without an error structure) is then fit and residuals from the model are examined, which suggest that an MA(1) structure should be added to correct for the autocorrelated errors. Figure 14 shows the specification for our final model:

Figure 14: Specify a Distributed Lag Model

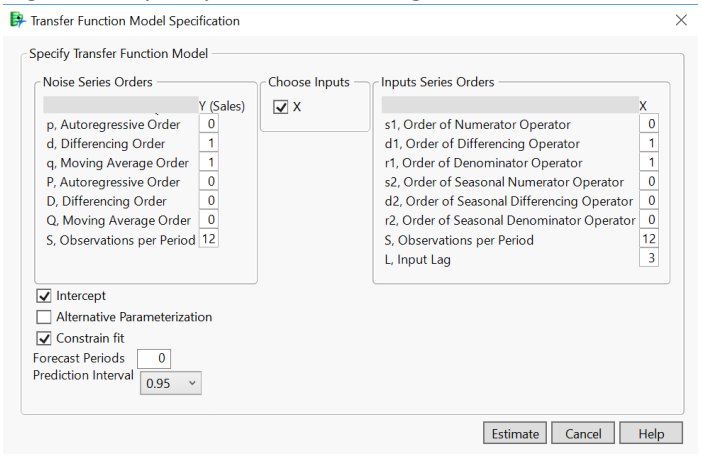


Figure 15 shows the estimation details and the final distributed lag model. (JMP reports from other steps not shown here are provided in the companion JMP Journal.)

Figure 15: Distributed Lag Model with MA(1) Error

Final Model

Model Summary

DF	117
Sum of Squared Errors	6.10748507
Variance Estimate	0.05220072
Standard Deviation	0.22847478
Akaike's 'A' Information Criterion	-9.5403984
Schwarz's Bayesian Criterion	1.64276382
RSquare	0.97876377
RSquare Adj	0.97823286
MAPE	0.2741003
MAE	0.56434373
-2LogLikelihood	-17.540398

Parameter Estimates

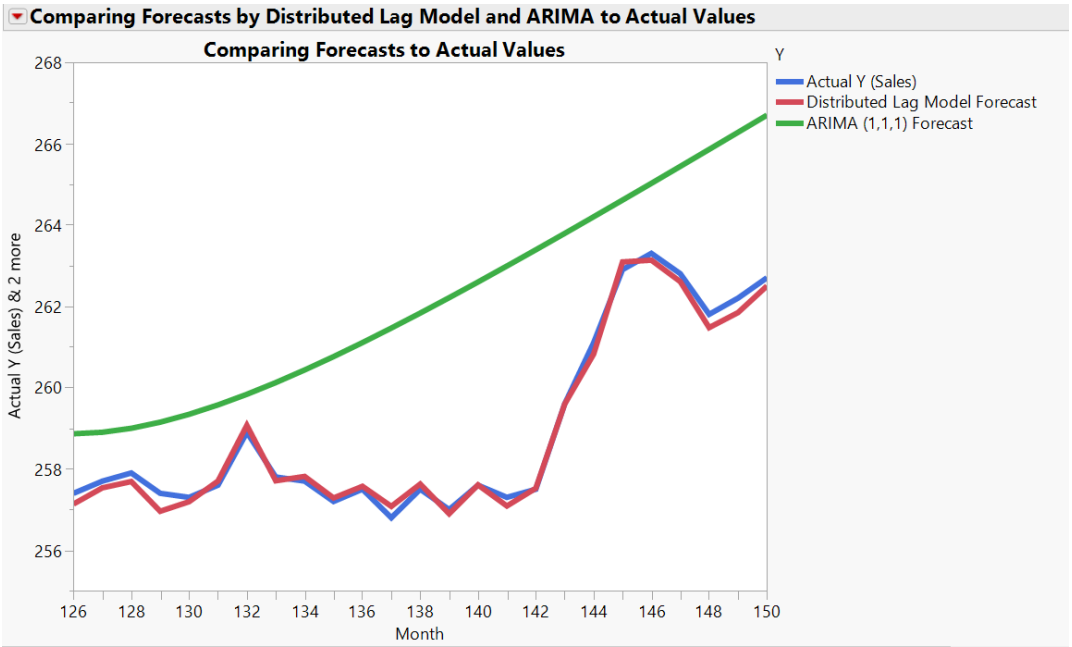
Variable	Term	Factor	Lag	Estimate	Std Error	t Ratio	Prob>|t|
X	Num0,0	0	0	4.7264416	0.0572476	82.56	<.0001*
X	Den1,1	1	1	0.7245734	0.0041548	174.39	<.0001*
Y (Sales)	MA1,1	1	1	0.5832683	0.0785985	7.42	<.0001*
	Intercept	0	0	0.0287036	0.0099918	2.87	0.0048*

$$(1-B)\cdot Y(\text{Sales})_t = 0.0287 + \left(\frac{4.7264}{(1-0.7246\cdot B)}\right)\cdot(1-B)\cdot X_{t-3} + (1-0.5833\cdot B)\cdot e_t$$

We now use the test data to forecast the sales with the actual values of x and compare the forecasts to the actual sales. For model comparison I've also selected a best ARIMA model, ARIMA (1,1,1), and generated the forecasts from this model.

Figure 16 shows the forecasts from these two time series models compared to the actual sales. Clearly, the Distributed Lag Model makes better forecasts than the ARIMA. This is not surprising, as we saw in the previous example, because the Distributed Lag Model captures both the systematic dynamics of the input variable and autocorrelations.

Figure 16: Model Comparison: Forecasts vs Actuals



Use Case 3: Intervention Models

Suppose an output time series (e.g., defects) is interrupted by inventions (e.g., changes in manufacturing processes). We can employ the Transfer Function Model to analyze and model such external shocks.

In fact, the intervention model is no different from the above distributed lag model (Eq. 7) except that x_t is an indicator series of 0s and 1s. Assume the change occurs at T, then

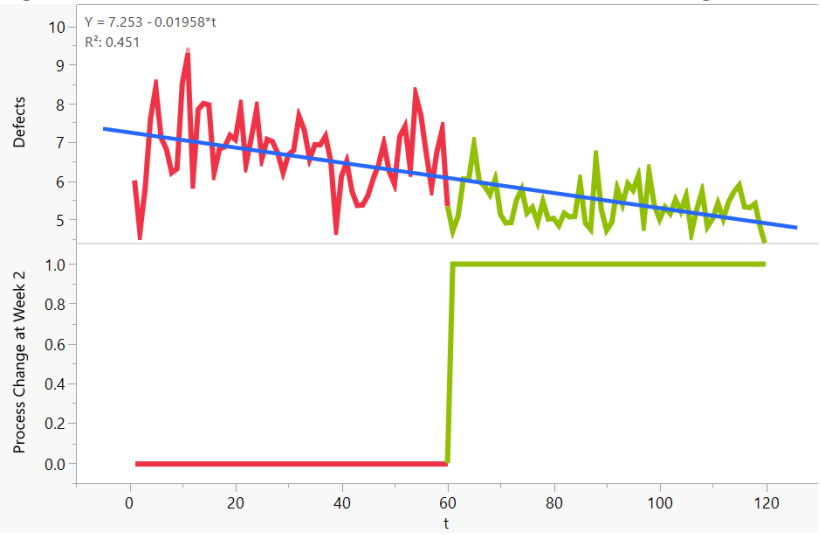
$$x_t = \begin{cases} 0 & t < T \\ 1 & t \geq T \end{cases}$$

Example 3: Defects and Process Changes[5]

Figure 16 shows the hourly monitoring data on defect rates of a manufacturing process. To improve the yield a new process is implemented at the beginning of Week2.

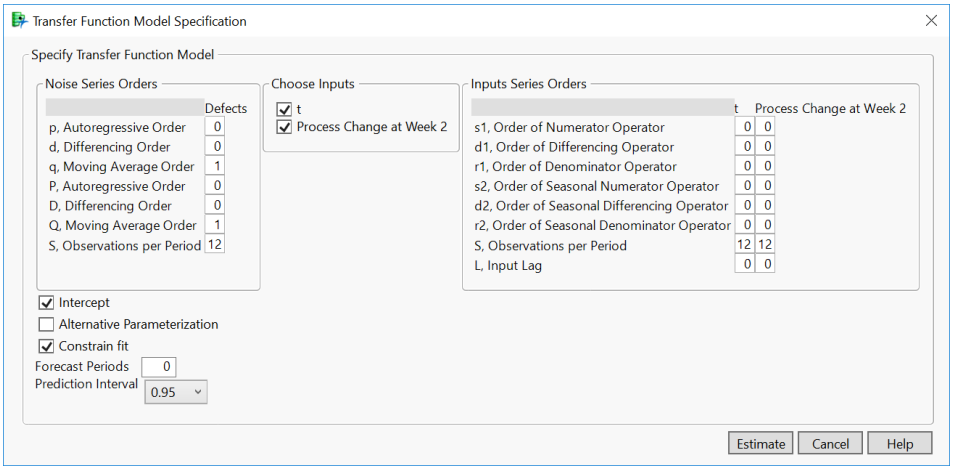
[5] The JMP reports are provided in the companion Journal.

Figure 16: Defect Rates: Before and After the Process Changes



The plot above shows there is an overall downward trend and a level shift in defect rate because of the process change. To model both the linear trend and interventions, I fit the transfer function model with two predictors: a linear trend, *t*, and an indicator variable, *Process Change at Week 2*, by OLS. After I check the residuals I re-fit the model with an ARMA error correction (Figure 17).

Figure 17: Specify an Intervention Model with MA1,1xMA2,12 Errors



Notice that the error structure (i.e., noise series) contains two components: one for the short-term moving average errors (*MA1,1)*, and the seocnd (*MA2,12)* for the 12-hour seasonalty apprently due to the fact the factory opearates on a 12-hour shift schedule The model and fit details are shown in Figure 18.

Figure 18: Intervention Models with MA1,1xMA2,12 Errors

Intervention Models

Model Summary

DF	115
Sum of Squared Errors	45.1850097
Variance Estimate	0.39291292
Standard Deviation	0.62682766
Akaike's 'A' Information Criterion	237.715705
Schwarz's Bayesian Criterion	251.653163
RSquare	0.63068385
RSquare Adj	0.61783807
MAPE	8.34387436
MAE	0.50377014
-2LogLikelihood	227.715706

Parameter Estimates

Variable	Term	Factor	Lag	Estimate	Std Error	t Ratio	Prob>|t|
t	Num0,0	0	0	-0.010057	0.0038556	-2.61	0.0103*
Process Change at Week 2	Num0,0	0	0	-0.780217	0.2611610	-2.99	0.0034*
Defects	MA1,1	1	1	-0.314208	0.0810772	-3.88	0.0002*
Defects	MA2,12	2	12	0.547387	0.0983989	5.56	<.0001*
	Intercept	0	0	7.054878	0.1271419	55.49	<.0001*

$$\text{Defects}_t = \left(\left(7.0549 - 0.0101 \bullet t_t\right) - 0.7802 \bullet \text{Process Change at Week 2}_t\right) + \left(1 + 0.3142 \bullet B\right) \bullet \left(1 - 0.5474 \bullet B^{12}\right) \bullet e_t$$

The estimation results show that the defect rate decreases by 0.01 every hour and that the new manaufactoring process brings down the defect rate by 0.78.

4. Summary

I have shown the versatile transfer function models through examples to demonstrate the benefits of using the transfer functions for building better forecasting models. Compared to the OLS regression and ARIMA models, the transfer function models produce better forecast.

As with any analysis using observational data the time series modeling is no exception that model effects may not be interpreted as causal or the cross-correlations might even be spurious. Use a test data to validate a model's out-of-sample predictions, or at a minimum, calculate a one-step-ahead forecast error for evaluations. Updating the model with new data improves the short-term forecasting accuracy. Lastly, conducting a designed experiment, if possible, is always the best approach to generate the data for analyzing the input-output relationship.

References

George Box, Gwilym Jenkins, Gregory Reinsel and Greta Ljung. 2016. *Time Series: Forecasting and Control*. 5th ed. Hoboken, NJ: John Wiley & Sons.

Rob J. Hyndman and George Athanasopoulos. 2018. *Forecasting: Principles and Practices* (2nd ed.). Melbourne, Australia: OTexts.org (Free one line at *https://otexts.com/fpp2/*)

SAS Institute Inc. 2018. *JMP® 14 Predictive and Specialized Modeling.* Cary, NC: SAS Institute Inc. (*https://www.jmp.com/support/help/14-2/time-series-analysis.shtml#*)