TRANSFER FUNCTIONS AND INTERVENTION MODELS

Outcome: Describe the transfer functions and intervention models

Introduction

- ARIMA models can be improved by introducing certain inputs reflecting these changes in the process conditions.
- This will lead to what is known as transfer function–noise models.

Transfer Function Models

- Transfer function models are used to understand the relationship between an input time series and an output time series.
- They are particularly useful when you have leading indicators or other exogenous variables that can help improve the accuracy of your forecasts.
- These models can be seen as an extension of ARIMA (Auto-Regressive Integrated Moving Average) models, incorporating external variables to better predict future values

Transfer Function Models

- **Key Components**
  - **Input Series**: The external variable that influences the output series.
  - **Output Series**: The main time series you are trying to forecast.
  - **Transfer Function**: Describes how the input series affects the output series over time.

Intervention Models

- Intervention models, also known as **interrupted time series analysis,** are used **to evaluate the impact of an intervention or event on a time series.**
- This could be a policy change, a new regulation, or any significant event that might disrupt the usual pattern of the data[2].
- **Definition: Intervention models are frameworks used to evaluate the impact of specific actions or policies aimed at changing an outcome.**
- They are commonly employed in social sciences, health interventions, and program evaluations.

- **Components**: An intervention model typically includes:
  - **Inputs**: Resources, actions, or strategies implemented.
  - **Outputs**: Immediate results or responses from the intervention.
  - **Outcomes**: Long-term effects or changes in behavior, health, or conditions.

- **Types**: Common types of intervention models include:
    - **Logic Models**: Visual representations that outline the relationship between resources, activities, outputs, and outcomes.
    - **Causal Models**: Used to understand the cause-and-effect relationships in interventions.

Intervention Models

- **Applications**
  - **Economics**: Evaluating the impact of policy changes on economic indicators.
  - **Healthcare**: Assessing the effect of new treatments or health policies.
  - **Marketing**: Measuring the impact of advertising campaigns on sales.

Transfer Function Models

- Transfer function models are a **powerful tool in time series analysis**, **used to model the relationship between an input (or exogenous) time series and an output (or dependent) time series**.

- **Key Components**
  - **Input Series (X)**: This is the external variable that influences the output series. For example, in an economic model, this could be a leading indicator like interest rates.
  - **Output Series (Y)**: This is the main time series you are trying to forecast, such as GDP or sales figures.
  - **Transfer Function**: This function describes how the input series affects the output series over time. It typically includes parameters that capture the delay and the magnitude of the effect.

- **Mathematical Representation**
- A simple transfer function model can be represented as:

$$Y_t = \sum_{i=0}^{q} \delta_i X_{t-i} + \sum_{j=1}^{p} \phi_j Y_{t-j} + \epsilon_t$$

- Where:
  - (Y_t) is the output series at time (t).
  - (X_{t-i}) is the input series at time (t-i).
  - (\delta_i) are the coefficients that measure the impact of the input series.
  - (\phi_j) are the coefficients for the autoregressive part of the model.
  - (\epsilon_t) is the error term.

Steps to Build a Transfer Function Model

1. **Identify the Input and Output Series**: Determine which series will be the input and which will be the output.
2. **Preliminary Analysis**: Conduct exploratory data analysis to understand the characteristics of both series.
3. **Model Identification**: Use techniques like cross-correlation to identify the appropriate lag structure.
4. **Parameter Estimation**: Estimate the parameters of the transfer function using methods like maximum likelihood estimation.
5. **Model Diagnostics**: Check the residuals of the model to ensure they behave like white noise.
6. **Forecasting**: Use the model to make forecasts of the output series based on future values of the input series.

Example: Transfer Function Model in R

Step 1: Install and Load Necessary Packages

First, ensure you have the necessary packages installed and loaded.

R

install.packages("forecast") install.packages("tseries") library(forecast) library(tseries)

Step 2: Load and Prepare Data

Load your data into R. For this example, let’s assume you have two time series: sales and ad_spend.

R

Example data sales <- ts(c(100, 120, 130, 150, 160, 180, 200, 220, 240, 260), frequency = 12) ad_spend <- ts(c(10, 15, 20, 25, 30, 35, 40, 45, 50, 55), frequency = 12)

Example: Transfer Function Model in R

Step 3: Identify the Transfer Function Model

Use cross-correlation to identify the relationship between the input and output series.

R

```
ccf(ad_spend, sales)
```

Step 4: Fit the Transfer Function Model

Fit the transfer function model using the Arima function from the forecast package.

R

```
# Fit ARIMA model with ad_spend as an external regressor model <-
Arima(sales, xreg = ad_spend, order = c(1, 0, 0)) summary(model)
```

Step 5: Forecast Using the Model

Use the fitted model to make forecasts.

R

```
# Forecast future sales with future ad_spend values future_ad_spend <- ts(c(60, 65, 70), frequency = 12) forecasted_sales <- forecast(model, xreg = future_ad_spend) plot(forecasted_sales)
```

Transfer Function–Noise Models

- Transfer function–**noise models are an extension of transfer function models that include a noise component to account for the unexplained variability in the output series.**
- These models are particularly useful when the relationship between the input and output series is not perfect, and there is some residual noise that needs to be modeled separately.

- **Key Components**
    - **Input Series (X)**: The external variable influencing the output series.
    - **Output Series (Y)**: The main time series you are trying to forecast.
    - **Transfer Function**: Describes how the input series affects the output series over time.
    - **Noise Component**: Captures the residual variability in the output series that is not explained by the input series.

- **Mathematical Representation**
- A transfer function–noise model can be represented as:

$$Y_t = \sum_{i=0}^{q} \delta_i X_{t-i} + \sum_{j=1}^{p} \phi_j Y_{t-j} + N_t$$

Where:

•(Y_t) is the output series at time (t).

•(X_{t-i}) is the input series at time (t-i).

•(\delta_i) are the coefficients that measure the impact of the input series.

•(\phi_j) are the coefficients for the autoregressive part of the model.

•(N_t) is the noise component, which can be modeled as an ARIMA process.

Steps to Build a Transfer Function–Noise Model

1. **Identify the Input and Output Series**: Determine which series will be the input and which will be the output.
2. **Preliminary Analysis**: Conduct exploratory data analysis to understand the characteristics of both series.
3. **Model Identification**: Use techniques like cross-correlation to identify the appropriate lag structure for the transfer function.
4. **Fit the Transfer Function**: Estimate the parameters of the transfer function.
5. **Model the Noise Component**: Fit an ARIMA model to the residuals (noise component) of the transfer function model.
6. **Combine Models**: Combine the transfer function and noise models to form the complete transfer function–noise model.
7. **Model Diagnostics**: Check the residuals of the combined model to ensure they behave like white noise.
8. **Forecasting**: Use the combined model to make forecasts of the output series based on future values of the input series.

Transfer Function–Noise Models in R using the tfarima package: Example

1.Install and Load the Package:

R

```
install.packages("tfarima") library(tfarima)
```

2. Prepare Your Data: Ensure your data is in a time series format. You can use the ts function to convert your data if needed.

R

```
data <- ts(your_data, start = c(Year, Month), frequency = 12) # Example for monthly data
```

Transfer Function–Noise Models in R using the tfarima package: Example

3. **Identify the Model**: Use the identifyTF function to identify the transfer function model.

R

```
identifyTF(data, input_series)
```

4. Estimate the Model: Use the estimateTF function to estimate the parameters of the transfer function model.

R

```
model <- estimateTF(data, input_series, order = c(p, d, q), seasonal =
list(order = c(P, D, Q), period = S))
```

Transfer Function–Noise Models in R using the tfarima package: Example

5. Diagnose the Model: Check the residuals of the model to ensure it fits well.

R

```
checkResiduals(model)
```

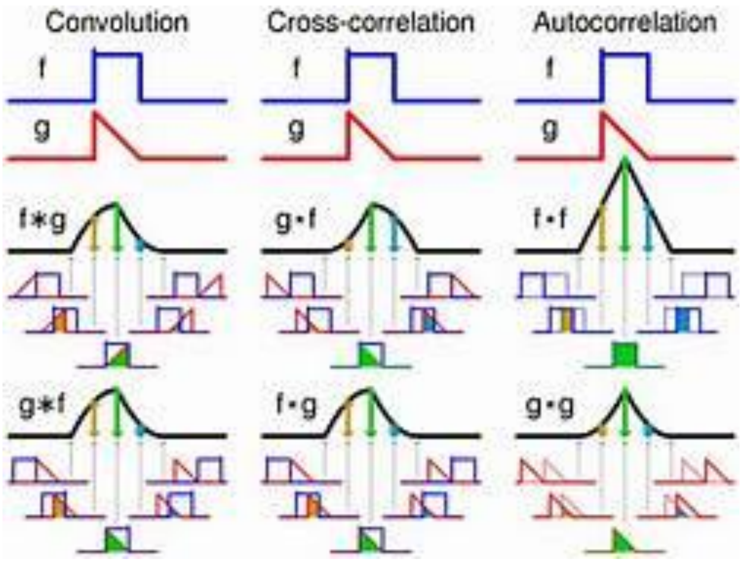
6. Forecast Using the Model: Use the forecastTF function to make predictions.

R

```
forecast <- forecastTF(model, h = 12) # Forecasting 12 periods ahead
plot(forecast)
```

Cross-correlation

- Cross-correlation is a statistical measure that describes the similarity between two time series as a function of the lag of one relative to the other.
- It’s often used in signal processing, pattern recognition, and time series analysis to find the degree to which two series are correlated.



- **Definition**
- For two time series (x(t)) and (y(t)), the cross-correlation function (R_{xy}(\tau)) at lag (\tau) is defined as:

$$R_{xy}(\tau) = \sum_{t} x(t) \cdot y(t + \tau)$$

- **Properties**
- **Symmetry**: (R_{xy}(\tau) = R_{yx}(-\tau))
- **Maximum Value**: The maximum value of the cross-correlation function indicates the point of highest similarity between the two series.
- **Normalization**: Often, the cross-correlation function is normalized to ensure the values lie between -1 and 1.

- **Applications**
- **Signal Processing**: To detect known patterns within a signal.
- **Econometrics**: To study the lead-lag relationships between economic indicators.
- **Neurophysiology**: To analyze the relationship between different neural signals.
- **System Identification**: Using time series transfer functions helps in identifying dynamic relationships between variables in fields like economics, engineering, and environmental science.
- **Lag Analysis**: Cross-correlation can reveal how the effect of an input variable (e.g., economic policy) influences an output variable (e.g., GDP) over time, which is crucial for timing interventions.

Example in R

Here's how you can compute and plot the cross-correlation function in R:

1. **Install and Load Necessary Packages**:

R

```
install.packages("stats") library(stats)
```

2. **Prepare Your Data**: Ensure your data is in a time series format.

R

```
ts1 <- ts(data1, start = c(2020, 1), frequency = 12) ts2 <- ts(data2, start = c(2020, 1), frequency = 12)
```

3. **Compute Cross-Correlation**: Use the ccf function to compute the cross-correlation.

R

```
ccf(ts1, ts2, lag.max = 20, plot = TRUE)
```

- **Example Workflow**
  - **Data Preparation**: Collect and preprocess the time series data.
  - **Estimation of Transfer Function**: Use statistical techniques (like ARIMA or state-space models) to estimate the transfer function.
  - **Cross-Correlation Analysis**: Compute the cross-correlation between the input and output series to identify significant lags.
  - **Interpretation**: Analyze the results to understand the causal relationships and the timing of effects.

Transfer Function–Noise Model Specification

- **Understanding the Components**
  - **Input Series (X(t)X(t)X(t))**: This is the variable that influences the output.
  - **Output Series (Y(t)Y(t)Y(t))**: This is the variable being predicted or analyzed.
  - **Transfer Function (H(B)H(B)H(B))**: This describes how the input affects the output over time.
  - **Noise/Error Term (ϵ(t)\epsilon(t)ϵ(t))**: Represents the random disturbances affecting the output.

Model Specification Steps:

1. **Preliminary Identification**:

   Impulse Response Coefficients: Identify the coefficients that describe how the input affects the output over time. This involves examining the cross-correlation function between the input and output series[1].

2. **Specification of the Noise Term**:

   Noise Model: Determine the appropriate noise model, often an ARIMA (AutoRegressive Integrated Moving Average) model, to account for the autocorrelation in the residuals[1].

3. **Specification of the Transfer Function**:

   Transfer Function Form: Choose the form of the transfer function, which could be a simple lagged relationship or a more complex distributed lag model[2].

4. **Estimation**:

   Parameter Estimation: Use statistical techniques to estimate the parameters of both the transfer function and the noise model. This often involves iterative methods to maximize the likelihood function[1].

5. **Model Diagnostic Checks**:

   Residual Analysis: Check the residuals of the model to ensure they behave like white noise, indicating a good fit. This involves examining autocorrelation and partial autocorrelation functions of the residuals[1].

- **Example Workflow**
  - **Identify the impulse response coefficients** by examining the cross-correlation between input and output.
  - **Specify the noise term** using an ARIMA model.
  - **Define the transfer function** based on the identified impulse response.
  - **Estimate the parameters** using maximum likelihood estimation.
  - **Perform diagnostic checks** to validate the model.

Forecasting with Transfer Function–Noise Models

- **Forecasting with Transfer Function–Noise Models**
- Forecasting using Transfer Function–Noise (TFN) models involves several steps to ensure accurate predictions. Here’s a detailed guide:
- **Model Identification**:
  - **Impulse Response Analysis**: Identify how the input series affects the output series over time. This involves examining the cross-correlation function between the input and output series to determine the lag structure.
- **Model Specification**:
  - **Transfer Function**: Specify the form of the transfer function, which could be a simple lagged relationship or a more complex distributed lag model.
  - **Noise Model**: Specify the noise model, typically an ARIMA model, to account for the autocorrelation in the residuals.

- **Parameter Estimation**:
  - **Estimate Parameters**: Use statistical techniques such as Maximum Likelihood Estimation (MLE) to estimate the parameters of both the transfer function and the noise model.
- **Model Validation**:
  - **Residual Analysis**: Check the residuals to ensure they behave like white noise, indicating a good fit. This involves examining the autocorrelation and partial autocorrelation functions of the residuals.
- **Forecasting**:
  - **Generate Forecasts**: Use the estimated model to generate forecasts. This involves applying the transfer function to the input series and adding the noise component.
  - **Confidence Intervals**: Calculate confidence intervals for the forecasts to quantify the uncertainty.

- **Example Workflow**
  - **Identify the impulse response coefficients** by examining the cross-correlation between input and output.
  - **Specify the noise term** using an ARIMA model.
  - **Define the transfer function** based on the identified impulse response.
  - **Estimate the parameters** using maximum likelihood estimation.
  - **Perform diagnostic checks** to validate the model.
  - **Generate forecasts** and calculate confidence intervals.

- **Practical Application**
- Let’s say you have a time series of sales data (output) and advertising spend (input). You can use a TFN model to forecast future sales based on past advertising spend. The steps would be:
- **Identify** the relationship between advertising spend and sales.
- **Specify** the transfer function and noise model.
- **Estimate** the parameters.
- **Validate** the model.
- **Forecast** future sales based on projected advertising spend.

Example:

- Example of how you can implement a Transfer Function–Noise (TFN) model in R for forecasting. This example assumes you have two time series: input_series (e.g., advertising spend) and output_series (e.g., sales).

Step-by-Step R Code Example

1.Load Necessary Libraries:

R

```
install.packages("forecast") install.packages("TSA") library(forecast)
library(TSA)
```

2. Prepare the Data:

R

```
# Example data input_series <- ts(c(1, 2, 3, 4, 5, 6, 7, 8, 9, 10), frequency = 12) output_series <- ts(c(2, 3, 5, 7, 11, 13, 17, 19, 23, 29), frequency = 12)
```

3. Identify the Transfer Function:

R

```
# Cross-correlation function to identify lags ccf(input_series, output_series)
```

4. Fit the Transfer Function–Noise Model:

R

```
# Fit the transfer function model
tf_model <- arimax(output_series, order = c(1, 0, 0), xtransf = input_series,
transfer = list(c(0, 1))) summary(tf_model)
```

5. Check Residuals:

R

```
# Check residuals to ensure they are white noise
tsdisplay(residuals(tf_model))
```

6. Forecasting:

R

```
# Forecast future values
forecast_horizon <- 12
future_input <- ts(c(11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22), frequency
= 12)
forecasts <- predict(tf_model, n.ahead = forecast_horizon, newxreg =
future_input)
plot(forecasts$pred, type = "l", col = "blue", ylim = range(c(output_series,
forecasts$pred)))
lines(output_series, col = "black")
```

Explanation

1. **Load Necessary Libraries**: Install and load the required libraries.
2. **Prepare the Data**: Create example time series data for input and output.
3. **Identify the Transfer Function**: Use the cross-correlation function to identify the relationship between input and output.
4. **Fit the Model**: Fit the TFN model using the `arimax` function.
5. **Check Residuals**: Ensure the residuals are white noise.
6. **Forecasting**: Generate forecasts for future periods and plot the results.

Intervention Analysis

- Intervention analysis is a statistical technique used to evaluate the impact of an intervention or event on a time series of data. This method is particularly useful in fields like economics, public health, and social sciences to assess the effectiveness of policies, treatments, or other interventions.

Intervention Analysis

- Here are the key steps involved in conducting an intervention analysis:
- **Identify the Intervention**: Determine the point in time when the intervention occurred.
- **Collect Data**: Gather time series data before and after the intervention.
- **Model the Data**: Use statistical models (e.g., ARIMA) to understand the underlying patterns in the data.
- **Estimate the Impact**: Assess the change in the time series data attributable to the intervention.
- **Validate the Results**: Check the robustness of the findings using diagnostic tests and validation techniques.

Intervention Analysis

- **Step-by-Step Example:**

1. Identify the Intervention:

- **Intervention**: Implementation of the new traffic law.
- **Date of Intervention**: January 1, 2023.

2. Collect Data:

- **Time Series Data**: Monthly number of road accidents from January 2020 to December 2023.

4. Model the Data:

- **Pre-Intervention Period**: January 2020 to December 2022.
- **Post-Intervention Period**: January 2023 to December 2023.
- Use an ARIMA model to fit the data and account for trends and seasonality.

5. Estimate the Impact:

- Compare the predicted number of accidents (based on the pre-intervention model) with the actual number of accidents after the intervention.
- Calculate the difference to estimate the impact of the new traffic law.

5. Validate the Results:

- Perform diagnostic checks on the model residuals to ensure they are randomly distributed.
- Use additional statistical tests (e.g., t-tests) to confirm the significance of the observed changes.

Example Data and Results:

Table

Month	**Actual Accidents**	**Predicted Accidents**	**Difference**
January 2023	80	100	-20
February 2023	75	95	-20
…	…	…	…
December 2023	60	90	-30

Let’s implement an intervention analysis in R to evaluate the impact of a new traffic law on the number of road accidents.

Step-by-Step Implementation in R:

1.Load the necessary libraries:

R

```
library(forecast)
library(tseries)
```

2. Simulate the data:

R

```
set.seed(123)
pre_intervention <- rnorm(36, mean = 100, sd = 10) # Data from Jan 2020 to Dec 2022
post_intervention <- rnorm(12, mean = 80, sd = 10) # Data from Jan 2023 to Dec 2023
accidents <- ts(c(pre_intervention, post_intervention), start = c(2020, 1), frequency = 12)
```

3. Plot the data:

R

```r
plot(accidents, main = "Monthly Road Accidents", ylab = "Number of
Accidents", xlab = "Time")
abline(v = 2023, col = "red", lwd = 2) # Mark the intervention point
```

4. Fit an ARIMA model to the pre-intervention data:

R

```
pre_intervention_data <- window(accidents, end = c(2022, 12))
fit <- auto.arima(pre_intervention_data)
summary(fit)
```

5. Forecast the post-intervention period:

R

```
forecasted <- forecast(fit, h = 12)
plot(forecasted)
lines(post_intervention, col = "red")
```

6. Compare the actual post-intervention data with the forecasted values:

R

```
actual_post <- window(accidents, start = c(2023, 1))
forecasted_values <- forecasted$mean
difference <- actual_post - forecasted_values
difference
```

Interpretation:

- The difference vector will show the difference between the actual and forecasted number of accidents for each month in the post-intervention period.
- Negative values indicate a reduction in accidents, suggesting the intervention was effective.

R Commands

Summary of Commands

1. **Load Libraries**: library(forecast) and library(ggplot2)
2. **Simulate Data**: Generate a synthetic time series dataset.
3. **Intervention Variable**: Create an intervention variable using ifelse().
4. **Fit Model**: Use Arima() to fit the model including the intervention.
5. **Check Residuals**: Use checkresiduals().
6. **Forecasting**: Use forecast() and plot with autoplot().