DLA

Clemens Glomb

Durbin-Levinson Algorithm

Introduction

In this vignette the application of the function 'DLA' is presented. DLA stands for the Durbin-Levinson algorithm, which is used for the recursive calculation of coefficients for time series data. The Durbin-Levinson algorithm is generally used for AR(p) processes. It enables the coefficients of autoregressive models to be calculated efficiently. This function is useful for modeling time series in order to make predictions. It can be used to understand the movement of the series over time.

Theoretical Background

The Durbin-Levinson algorithm works recursion-based and calculates the coefficients of an autoregressive (AR) model of a time series. It works as follows: For a given stationary time series $\{x_1, \dots, x_n\}$, the algorithm calculates the coefficients $\phi_{n1}, \dots, \phi_{nn}$ and the corresponding variances v_n.

The algorithm uses the following recursive steps to calculate the coefficient. At the beginning ϕ_{11} will be computed with $\frac{\gamma(1)}{\gamma(0)}$, while $\gamma(n)$ is the autocovariance of the time series.

$$\phi_{nn} = \left[\gamma(n) - \sum_{j=1}^{n-1} \phi_{n-1,j}\gamma(n-j)\right] v_{n-1}^{-1},$$

$$\begin{bmatrix} \phi_{n1} \\ \vdots \\ \phi_{n,n-1} \end{bmatrix} = \begin{bmatrix} \phi_{n-1,1} \\ \vdots \\ \phi_{n-1,n-1} \end{bmatrix} - \phi_{nn} \begin{bmatrix} \phi_{n-1,n-1} \\ \vdots \\ \phi_{n-1,1} \end{bmatrix},$$

$$v_n = v_{n-1}\left[1 - \phi_{nn}^2\right],$$

How the Function Works

Function Parameters The `DLA` function takes the following inputs:

- **X**: A numeric vector that represents the time series data. This time series should be stationary.
- **m**: The number of coefficients to be calculated. This parameter is determined after calling `DLA`.

Function Output The function returns a list containing the following elements:

- **phi**: A numeric vector of length `m`. It contains the calculated coefficients $\phi_{n1}, \ldots, \phi_{nn}$. Each element in `phi` represents the influence of previous values in the time series on the current value. For example, ϕ_{n1} shows how much the first lag (previous value) affects the current value of the time series. These coefficients help to understand the dependency structure within the time series.
- **v**: A numeric vector of length `m`. It contains the mean squared errors v_n. These variances quantify the uncertainty or error associated with each prediction in the AR model. Lower values in `v` indicate that the model's predictions are more certain, while higher values suggest more uncertainty. Essentially, `v` helps in understanding the reliability of the model at each step.

Practical Application

Simple Example: We have a simple time series: `c(2, 4, 6, 8, 10)`. We use the DLA function from the zeitreihen package to calculate the coefficients `phi` and the mean squared errors `v` for this time series, focusing on `m = 2`, which means we're interested in the first two coefficients and their corresponding mean squared errors.

```r
library(zeitreihen)
DLA_data <- zeitreihen::DLA(c(2, 4, 6, 8, 10))
```

```
## Warning in zeitreihen::DLA(c(2, 4, 6, 8, 10)): Please note: This algorithm works for stationary time
## For any other time series, the results may be incorrect.
```

```r
DLA_data(2)
```

```
## $phi
## [1]  0.5238095 -0.3095238
##
## $nu
## [1] 8.00000 6.72000 6.07619
```

Output Interpretation:

- `phi` (Coefficients): The output gives us two coefficients: `[1] 0.5238095 -0.3095238`.
  - **0.5238095**: This is the first AR coefficient, ϕ_{n1}. It suggests that about 52% of the value of the time series at each point is influenced by the previous value (lag 1).
  - **-0.3095238**: This is the second AR coefficient, ϕ_{n2}. It indicates that the value of the time series is negatively influenced (about -31%) by the value two steps back (lag 2). These coefficients help describe how past values in the series contribute to the current value.
- `v` (mean squared errors): The output for `v` is `[1] 8.00 6.72`.
  - **8.00**: This is the mean squared error for the first step. It represents the uncertainty or error when predicting the time series at the first lag. A variance of 8.00 suggests there is some variability in the predictions.
  - **6.72**: This is the mean squared error for the second step. It represents the uncertainty when predicting the time series at the second lag. The lower variance compared to the first step (6.72 vs. 8.00) suggests that the prediction at this step is slightly more reliable.

Realistic Example: Let's work with the following time series data:

```
set.seed(123)
data <- c(2, 4, 6, 8, 10)
```

We want to calculate the coefficients for this time series using the DLA function. First, we will create the DLA model object, and then we will calculate the coefficients for m = 2:

```
model <- DLA(data)
```

```
## Warning in DLA(data): Please note: This algorithm works for stationary time series with zero-mean.
## For any other time series, the results may be incorrect.
```

```
result <- model(2)
```

Here's what happens step by step:

1. **Start**: The algorithm begins by calculating the first coefficient ϕ_{11}. This is computed as $\gamma(1)/\gamma(0)$, where $\gamma(1)$ is the autocovariance and $\gamma(0)$ is the variance of the time series.
2. **Recursive Calculation**:For each step n, the algorithm calculates the coefficient ϕ_{nn} by adjusting the autocovariance $\gamma(n)$. This adjustment is based on the previous coefficients $\phi_{n-1,j}$ and the autocovariances $\gamma(n-j)$.
3. **Matrix Update**: The coefficients $\phi_{n-1,1}$ through $\phi_{n-1,n-1}$ are updated and the new value ϕ_{nn} is added. This is necessary to compute the next coefficients.
4. **mean squared errors**: The mean squared error v_n is calculated at each step to capture the uncertainty in the estimates. This variance is based on the previous variance v_{n-1} and the current coefficient ϕ_{nn}.
5. **Numerical Stability**: The algorithm continuously checks if the calculated variances v_n or $\gamma(0)$ are close to zero. If this happens, the process is stopped to avoid numerical issues.
6. **Results**: At the end, the algorithm returns two vectors: `phi`, containing the calculated coefficients, and `v`, containing the mean squared errors. These values are important for modeling and forecasting time series data.

Heavy Example: Let's work with the following time series data:

```
set.seed(456)
new_data <- rnorm(100)
```

We calculate the coefficients for this time series for m = 5:

```
model_new <- DLA(new_data)
```

```
## Warning in DLA(new_data): Please note: This algorithm works for stationary time series with zero-mea
## For any other time series, the results may be incorrect.
```

```
result_new <- model_new(5)
```

Then we can have a look on the coefficients

```r
print(result_new)
```

```
## $phi
## [1]  0.040976672 -0.037395399 -0.002937863 -0.159128062  0.006837763
##
## $nu
## [1] 0.9930957 0.9914715 0.9904116 0.9903169 0.9653263 0.9652811
```

Example 4: Incorrect inputs

If the input do not correspond to the correct format, appropriate error messages are printed to inform the user what is wrong with their input. Here are several examples of how incorrect inputs are handled:

The time series data X must be passed in an atomic vector:

```r
X <- list(1, 2, 3, 4, 5)
model <- DLA(X)
```

```
## Error in DLA(X): X must be numeric or complex atomic vector
```

Also, the values of X must be numeric or complex:

```r
X <- c(1, 2, "3", 4, "5i")
model <- DLA(X)
```

```
## Error in DLA(X): X must be numeric or complex atomic vector
```

In addition, the atomic vector may not contain any NAs:

```r
X <- c(1, NA, 2, 3, 4)
model <- DLA(X)
```

```
## Error in DLA(X): X may not contain NAs
```

The atomic vector may not contain Inf or -Inf values:

```r
X <- c(1, 2, Inf, 3, 4)
model <- DLA(X)
```

```
## Error in DLA(X): X may not contain Inf or -Inf values
```

The m paramter must be a value of length 1:

```r
X <- rnorm(100)
model <- DLA(X)
result <- model(c(1,2))
```

```
## Error in model(c(1, 2)): m must be a value of length 1
```

The m parameter must be an integer value between 0 and the length of X:

```r
X <- rnorm(100)
model <- DLA(X)
result <- model(101)
```

```
## Error in model(101): m must be between 0 and length of X
```

References

If you need more information, please refer to the package documentation or the following resources:

- Introduction to Time Series and Forecasting

Erweiterte Themen

- **Erweiterte Verwendung**: Diskutiere fortgeschrittene Anwendungsfälle der Funktion.
- **Vergleich mit anderen Methoden**: Vergleiche die Funktion mit alternativen Ansätzen oder Methoden.
- **Tipps und Tricks**: Gebe nützliche Hinweise zur Optimierung und Vermeidung von Fehlern.