Time Series

Time Series is a sequence of data points organized in time order.

The sequence captures data at equally spaced points in time. Data that is not collected regularly at equally spaced points is not considered time series.

Time Series Motivation

For most forecasting exercises, standard regression approaches do not work for Time Series models, mostly because:

- The data is correlated over time
- The data is often non-stationary, which is hard to model using regressions
- You need a lot of data for a forecast

Forecasting Problems

These are the two types of forecasting problems. Consider that the vast majority of applications employ univariate models, harder to combine variables when using time series data.

1. Univariate

Think of single data series containing of:

- Continuous data, binary data, or categorical data
- Multiple unrelated series
- Conditional series

2. Panel or Multivariate

Think of multiple related series identifying groups such as customer types, department or channel, or geographic joint estimation across series

Time Series Applications

Time series data is common across many industries. For example:

- Finance: stock prices, asset prices, macroeconomic factors
- E-Commerce: page views, new users, searches

- Business: transactions, revenue, inventory levels

Time series methods are used to:

- Understand the processes driving observed data
- Fit models to monitor or forecast a process
- Understand what influences future results of various series
- Anticipate events that require management intervention

Time Series Components

A time series can be decomposed into several components:

Trend – long term direction

Seasonality – periodic behavior

Residual – irregular fluctuations

Generally, models perform better if we can first remove known sources of variation such as trend and seasonality. The main motivation for doing decomposition is to improve model performance. Usually we try to identify the known sources and remove them, leaving resulting series (residuals) that we can fit against time series models

Decomposition Models

These are the main models to decompose Time Series components:

– Additive Decomposition Model

Additive models assume the observed time series is the sum of its components.

i.e. Observation = Trend + Seasonality + Residual

These models are used when the magnitudes of the seasonal and residual values are independent of trend.

– Multiplicative Decomposition Model

Multiplicative models assume the observed time series is the product of its components.

i.e. Observation = Trend * Seasonality * Residual

A multiplicative model can be transformed to an additive by applying a log transformation:

log(Time*Seasonality*Residual) = log(Time) + log(Seasonality) + log(Residual)

These models are used if the magnitudes of the seasonal and residual values fluctuate with trend.

– Pseudo-additive Decomposition Model

Pseudo-additive models combine elements of the additive and multiplicative models.

They can be useful when:

Time series values are close to or equal to zero.

We expect features related to a multiplicative model.

A division by zero needs to be solved in the form: Ot = Tt + Tt(St – 1) + Tt(Rt – 1) = Tt(St + Rt – 1)

Decomposition of time series allows us to remove deterministic components, which would otherwise complicate modeling.

After removing these components, the main focus is to model the residual.

Other Methods

These are some other approaches of time series decomposition:

- Exponential smoothing
- Locally Estimated Scatterplot Smoothing (LOESS)
- Frequency-based methods

Stationarity

Stationarity impacts our ability to model and forecast

- A stationary series has the same mean and variance over time
- Non-stationary series are much harder to model

Common approach:

- Identify sources of non-stationarity
- Transform series to make it stationary
- Build models with stationary series

The Augmented Dickey-Fuller (ADF) test specifically tests for stationarity.

- It is a hypothesis test: the test returns a p-value, and we generally say the series is non-stationary if the p-value is less than 0.05.
- It is a less appropriate test to use with small datasets, or data with heteroscedasticity (different variance across observations) present.
- It is best to pair ADF with other techniques such as: run-sequence plots, summary statistics, or histograms.

Common Transformations for Time Series include:

Transformations allow us to generate stationary inputs required by most models.

There are several ways to transform nonstationary time series data:

- Remove trend (constant mean)
- Remove heteroscedasticity with log (constant variance)
- Remove autocorrelation with differencing (exploit constant structure)
- Remove seasonality (no periodic component)
- Multiple transformations are often required.

Time Series Smoothing

Smoothing is a process that often improves our ability to forecast series by reducing the impact of noise.

There are many ways to smooth data. Some examples:

- Simple average smoothing
- Equally weighted moving average
- Exponentially weighted moving average

This are some suggestions for selecting a Smoothing Technique. If your data:

– lack a trend

    - Then use Single Exponential Smoothing

- have trend but no seasonality
    - Then use Double Exponential Smoothing

- have trend and seasonality
    - Then use Triple Exponential Smoothing

ARMA Models

ARMA models combine two models:

- The first is an autoregressive (AR) model. Autoregressive models anticipate series' dependence on its own past values.
- The second is a moving average (MA) model. Moving average models anticipate series' dependence on past forecast errors.
- The combination (ARMA) is also known as the Box - Jenkins approach.

ARMA models are often expressed using orders *p* and *q* for the *AR* and *MA* components.

For a time series variable *X* that we want to predict for time *t*, the last few observations are:

X_{t-3},\;X_{t-2},\;X_{t-1}

$$X_{t-3}, X_{t-2}, X_{t-1}$$

- *AR(p)* models are assumed to depend on the *last p values* of the time series. For *p=2,* the forecast has the form:

X_t=\phi_1X_{t-1}+\phi_2X_{t-2}+\omega_t

$$X_t=\phi_1 X_{t-1}+\phi_2 X_{t-2}+\omega_t$$

Here,

\omega_t

ω_t

is the *forecast error*;

\phi_1

ϕ

1

and

\phi_2

ϕ

2

are the (*p*=2) parameters (estimated by regression).

- *MA(q)* models are assumed to depend on the *last q values* of the forecast error. For *q*=2*,* the forecast has the form:

X_t=\theta_2\omega_{t-2}+\theta_1\omega_{t-1}+\omega_t

X

t

$=\theta$

2

ω

$t-2$

$+\theta$

1

ω

$t-1$

$+\omega$

t

Here,

\omega_t

ω

t

is the forecast error,

\omega_{t-1}

ω

$t-1$

is the previous forecast error, etc.

\theta_1

θ

1

and

\theta_2

θ

2

are the (*q*=*2*) parameters.

Combining the *AR(p)* and *MA(q)* models yields the *ARMA(p, q)* model. For *p=2*, *q=2,* the ARMA(2, 2) forecast has the form:

X_t=\phi_1X_{t-1}+\phi_2X_{t-2}+\theta_2\omega_{t-2}+\theta_1\omega_{t-1}+\omega_t

X

t

$=\phi$

1

X

$t-1$

$+\phi$

2

X

$t-2$

$+\theta$

2

ω

$t-2$

$+\theta$

1

ω

$t-1$

$+\omega$

t

\omega_t

ω

t

is the forecast error,

\phi_1

ϕ

1

,

\phi_2

ϕ

2

,

\theta_1

θ

1

, and

\theta_2

θ

2

are the (*p* + *q* = *4*) parameters.

ARMA Models Considerations

These are important considerations to keep in mind when dealing with ARMA models:

- The time series is assumed to be stationary.
- A good rule of thumb is to have at least 100 observations when fitting an ARMA model.

There are three stages in building an ARMA model:

Identification

At this stage you:

- Validate that the time series is stationary.
- Confirm whether the time series contains a seasonal component.

You can determine if seasonality is present by using autocorrelation and partial autocorrelation plots, seasonal subseries plots, and intuition (possible in some cases, i.e. seasonal sales of consumer products, holidays, etc.).

An Autocorrelation Plot is commonly used to detect dependence on prior observations.

It summarizes total (2-way) correlation between the variable and its past values.

The Partial Autocorrelation Plot also summarizes dependence on past observations.

However, it measures partial results (including all lags)

Seasonal Subseries Plot is one approach for measuring seasonality. This chart shows the average level for each seasonal period and illustrates how individual observations relate to this level.

Estimation

Once we have a stationary series, we can estimate AR and MA models. We need to determine *p* and *q*, the order of the AR and MA models.

One approach here is to look at autocorrelation and partial autocorrelation plots. Another approach is to treat *p* and *q* as hyperparameters and apply standard approaches (grid search, cross validation, etc.)

How do we determine the order p of the AR model?

- Plot confidence intervals on the Partial Autocorrelation Plot.
- Choose lag *p* such that partial autocorrelation becomes insignificant for *p + 1* and beyond

How can we determine the order *q* of the MA model?

- Plot confidence intervals on the Autocorrelation Plot
- Choose lag *q* such that autocorrelation becomes insignificant for *q + 1* and beyond.

Evaluation

You can assess your ARMA model by making sure that the residuals will approximate a Gaussian distribution (aka white noise). Otherwise, you need to iterate to obtain a better model.

These are guidelines to choose between an AR and a MA model based on the shape of the autocorrelation and partial autocorrelation plots.

SHAPE	**MODEL**
Exponential Decaying to zero	AR models
Alternating positive and negative decaying to zero	AR models

One or more spikes, the rest are close to zero	MA model
Decay after a few lags	Mixed AR and MA
All zero or close to zero	Data is random
High values at fixed intervals	Include seasonal AR term
No decay to zero	Series is not stationary

ARIMA Models

ARIMA stands for Auto-Regressive Integrated Moving Average.

ARIMA models have three components:

- AR Model
- Integrated Component
- MA Model

SARIMA Models

SARIMA is short for Seasonal ARIMA, an extension of ARIMA models to address seasonality.

This model is used to remove seasonal components.

- The SARIMA model is denoted SARIMA (p, d, q) (P, D, Q).
- P, D, Q represent the same as p, d, q but they are applied across a season.
- M = one season

ARIMA and SARIMA Estimation

These are the steps to estimate p, d, q and P, D, Q?

- Visually inspect a run sequence plot for trend and seasonality.
- Generate an ACF Plot.
- Generate a PACF Plot.
- Treat as hyperparameters (cross validate).

- Examine *information criteria* (*AIC*, *BIC*) which penalize the number of parameters the model uses.

ARMA Models

ARMA models combine two models:

- The first is an autoregressive (AR) model. Autoregressive models anticipate series' dependence on its own past values.
- The second is a moving average (MA) model. Moving average models anticipate series' dependence on past forecast errors.
- The combination (ARMA) is also known as the Box - Jenkins approach.

ARMA models are often expressed using orders *p* and *q* for the *AR* and *MA* components.

For a time series variable *X* that we want to predict for time *t*, the last few observations are:

$$X_{t-3},\;X_{t-2},\;X_{t-1}$$

X

$t-3$

$,X$

$t-2$

$,X$

$t-1$

- *AR(p)* models are assumed to depend on the *last p values* of the time series. For *p=2,* the forecast has the form:

$$X_t=\phi_1X_{t-1}+\phi_2X_{t-2}+\omega_t$$

X

t

$$= \phi_1 X_{t-1} + \phi_2 X_{t-2} + \omega_t$$

Here,

\omega_t

ω_t

is the *forecast error*;

\phi_1

ϕ_1

and

\phi_2

ϕ

2

are the (*p*=2) parameters (estimated by regression).

- *MA(q)* models are assumed to depend on the *last q values* of the forecast error. For *q*=2*,* the forecast has the form:

X_t=\theta_2\omega_{t-2}+\theta_1\omega_{t-1}+\omega_t

X

t

$=\theta$

2

ω

$t-2$

$+\theta$

1

ω

$t-1$

$+\omega$

t

Here,

\omega_t

ω

t

is the forecast error,

\omega_{t-1}

ω

$t-1$

is the previous forecast error, etc.

\theta_1

θ

1

and

\theta_2

θ

2

are the (*q*=2) parameters.

Combining the *AR(p)* and *MA(q)* models yields the *ARMA(p, q)* model. For *p*=2, *q=2,* the ARMA(2, 2) forecast has the form:

X_t=\phi_1X_{t-1}+\phi_2X_{t-2}+\theta_2\omega_{t-2}+\theta_1\omega_{t-1}+\omega_t

$$X_t = \phi_1 X_{t-1} + \phi_2 X_{t-2} + \theta_2 \omega_{t-2} + \theta_1 \omega_{t-1}$$

$+\omega$

t

\omega_t

ω

t

is the forecast error,

\phi_1

ϕ

1

,

\phi_2

ϕ

2

,

\theta_1

θ

1

, and

\theta_2

θ

are the (*p* + *q* = *4*) parameters.

ARMA Models Considerations

These are important considerations to keep in mind when dealing with ARMA models:

- The time series is assumed to be stationary.
- A good rule of thumb is to have at least 100 observations when fitting an ARMA model.

There are three stages in building an ARMA model:

Identification

At this stage you:

- Validate that the time series is stationary.
- Confirm whether the time series contains a seasonal component.

You can determine if seasonality is present by using autocorrelation and partial autocorrelation plots, seasonal subseries plots, and intuition (possible in some cases, i.e. seasonal sales of consumer products, holidays, etc.).

An Autocorrelation Plot is commonly used to detect dependence on prior observations.

It summarizes total (2-way) correlation between the variable and its past values.

The Partial Autocorrelation Plot also summarizes dependence on past observations.

However, it measures partial results (including all lags)

Seasonal Subseries Plot is one approach for measuring seasonality. This chart shows the average level for each seasonal period and illustrates how individual observations relate to this level.

Estimation

Once we have a stationary series, we can estimate AR and MA models. We need to determine *p* and *q*, the order of the AR and MA models.

One approach here is to look at autocorrelation and partial autocorrelation plots. Another approach is to treat *p* and *q* as hyperparameters and apply standard approaches (grid search, cross validation, etc.)

How do we determine the order p of the AR model?

- Plot confidence intervals on the Partial Autocorrelation Plot.
- Choose lag *p* such that partial autocorrelation becomes insignificant for *p + 1* and beyond

How can we determine the order *q* of the MA model?

- Plot confidence intervals on the Autocorrelation Plot
- Choose lag *q* such that autocorrelation becomes insignificant for *q + 1* and beyond.

Evaluation

You can assess your ARMA model by making sure that the residuals will approximate a Gaussian distribution (aka white noise). Otherwise, you need to iterate to obtain a better model.

These are guidelines to choose between an AR and a MA model based on the shape of the autocorrelation and partial autocorrelation plots.

SHAPE	**MODEL**
Exponential Decaying to zero	AR models
Alternating positive and negative decaying to zero	AR models
One or more spikes, the rest are close to zero	MA model
Decay after a few lags	Mixed AR and MA
All zero or close to zero	Data is random

High values at fixed intervals	Include seasonal AR term
No decay to zero	Series is not stationary

ARIMA Models

ARIMA stands for Auto-Regressive Integrated Moving Average.

ARIMA models have three components:

- AR Model
- Integrated Component
- MA Model

SARIMA Models

SARIMA is short for Seasonal ARIMA, an extension of ARIMA models to address seasonality.

This model is used to remove seasonal components.

- The SARIMA model is denoted SARIMA (p, d, q) (P, D, Q).
- P, D, Q represent the same as p, d, q but they are applied across a season.
- M = one season

ARIMA and SARIMA Estimation

These are the steps to estimate p, d, q and P, D, Q?

- Visually inspect a run sequence plot for trend and seasonality.
- Generate an ACF Plot.
- Generate a PACF Plot.
- Treat as hyperparameters (cross validate).
- Examine *information criteria* (*AIC*, *BIC*) which penalize the number of parameters the model uses.

Deep Learning for Time Series Forecasting

Neural networks offer several benefits over traditional time series forecasting models, including:

- Automatically learn how to incorporate series characteristics like trend, seasonality, and autocorrelation into predictions.
- Able to capture very complex patterns.
- Can simultaneously model many related series instead of treating each separately.

Some disadvantages of using Deep Learning for Time Series Forecasting are:

- Models can be complex and computationally expensive to build (GPUs can help).
- Deep Learning models often overfit.
- It is challenging to explain / interpret predictions made by the model (“black box”).
- Tend to perform best with large training datasets.

Recurrent neural networks (RNNs) map a sequence of inputs to predicted output(s).

- Most common format is “many-to-one”, that maps an input sequence to one output value.
- Input at each time step sequentially updates the RNN cell’s “hidden state” (“memory”).
- After processing the input sequence, the hidden state information is used to predict the output.

RNNs often struggle to process long input sequences. It is mathematically difficult for RNNs to capture long-term dependencies over many time steps, which is a problem for Time Series, as sequences are often hundreds of steps. Another type of Neural Networks, Long short-term memory networks (LSTMs) can mitigate these issues with a better memory system

Long short-term memory networks share RNNs’ conceptual structure.

- LSTM cells have the same role as RNN cells in sequential processing of the input sequence.
- LSTM cells are internally more complex, with gating mechanisms and two states: a hidden state and a cell state.

Long short-term memory networks regulate information flow and memory storage.

- LSTM cells share forget, input, and output gates that control how memory states are updated and information is passed forward.
- At each time step, the input and current states determine the gate computations.

LSTMs vs RNNs

LSTMs are better suited for handling long-term dependencies than RNNs. However, they are much more complex, requiring many more trainable weights. As a result, LSTMs tend to take longer to train (slower backpropagation) and can be more prone to overfitting.

These are some guidelines on how to choose LSTMs or RNNs in a Forecasting task:

Always consider the problem at hand:

- If sequences are many time steps long, an RNN may perform poorly.
- If training time is an issue, using a LSTM may be too cumbersome.
- Graphics processing units (GPUs) speed up all neural network training, but are especially recommended when training LSTMs on large datasets.

Survival Analysis

Survival Analysis focuses on estimating the length of time until an event occurs. It is called 'survival analysis' because it was largely developed by medical researchers interested in estimating the expected lifetime of different cohorts. Today, these methods are applied to many types of events in the business domain.

Examples:

- How long will a customer remains on books before churning
- How long until equipment needs repairs

Survival Analysis is useful when we want to measure the risk of events occurring and our data are Censored.

- This can be referred to as failure time, event time, or survival time.
- If our data are complete and unbiased, standard regression methods may work.
- Survival Analysis allows us to consider cases with incomplete or censored data.

The Survival Function is defined as

S(t)=P(T>t)

$S(t)=P(T>t)$. Itmeasures the probability that a subject will survive past time *t*.

This function:

- Is decreasing (non-increasing) over time.
- Starts at 1 for all observations when *t=0*

- Ends at 0 for a high-enough *t*

The Hazard Rate is defined as:

h(t)=\frac{f(t)}{S(t)}

$h(t)=$

$S(t)$

$f(t)$

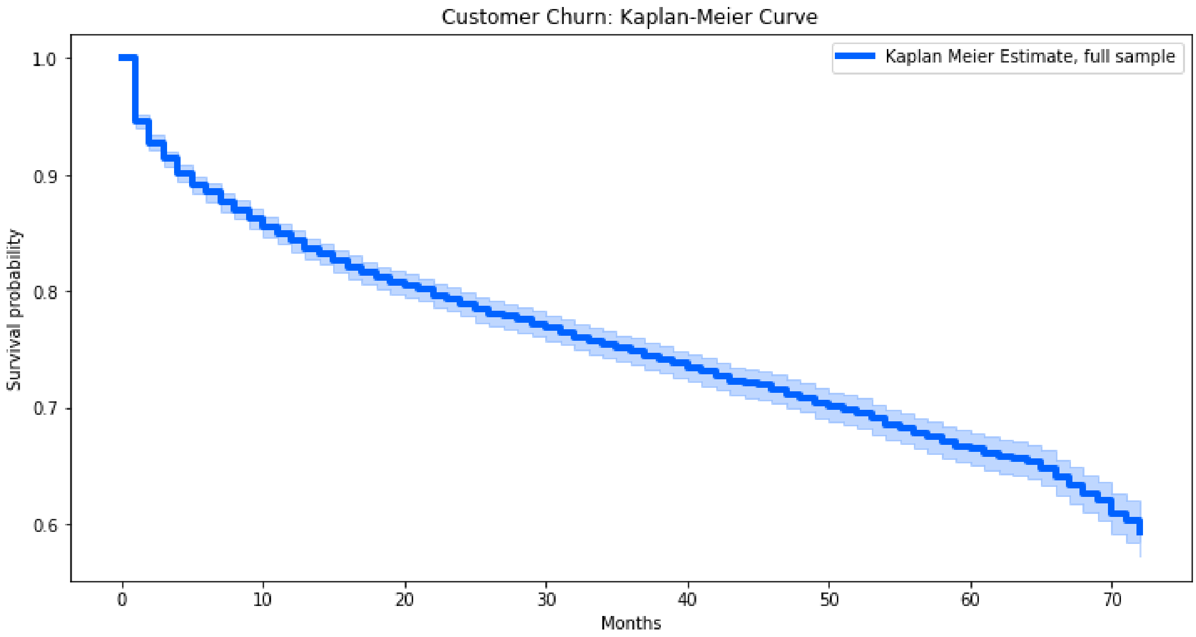
It represents the instantaneous rate at which events occur, given that it has not occurred already.

The cumulative hazard rate (sum of

h(t)

$h(t)$ from *t = 0* to *t = t*) represents accumulated risk over time.

The Kaplan-Meier estimator is a non-parametric estimator. It allows us to use observed data to estimate the survival distribution. The Kaplan-Meier Curve plots the cumulative probability of survival beyond each given time period.



Using the Kaplan-Meier Curve allows us to visually inspect differences in survival rates by category. We can use Kaplan-Meier Curves to examine whether there appear to be differences based on this feature.

To see whether survival rates differ based on number of services, we estimate Kaplan-Meier curves for different groups.

Survival Analysis Approaches

The Kaplan-Meier approach provides sample averages. However, we may want to make use of individual-level data to predict survival rates.

Some well-known Survival models for estimating Hazard Rates include these Survival Regression approaches. These methods:

- Allow us to generate estimates of total risk as a function of time
- Make use of censored and uncensored observations to predict hazard rates
- Allow us to estimate feature effects

Although these methods use time, these methods are not generally predicting a time to an event, rather predicting survival risk (or hazard risk) as a function of time.

– The Cox Proportional Hazard (CPH) model

This is one of the most common survival models. It assumes features have a constant proportional impact on the hazard rate.

For a single non-time-varying feature *X*, the hazard rate

h(t)

$h(t)$ is modeled as:

h(t)=\beta_0(t)e^{\beta_1X}

$h(t)=\beta$

0

$(t)e$

β

1

X

\beta_0(t)

β

0

(t) is the time-varying *baseline hazard,* and e^{\beta_1X}

e

β

1

X

is the (constant) proportional adjustment to the baseline hazard due to *X*.

Using the CPH model, we can plot estimated survival curves for various categories.

– Accelerated Failure Time (AFT) models (several variants including the Weibull AFT model)

These models differ with respect to assumptions they make about the hazard rate function, and the impact of features.