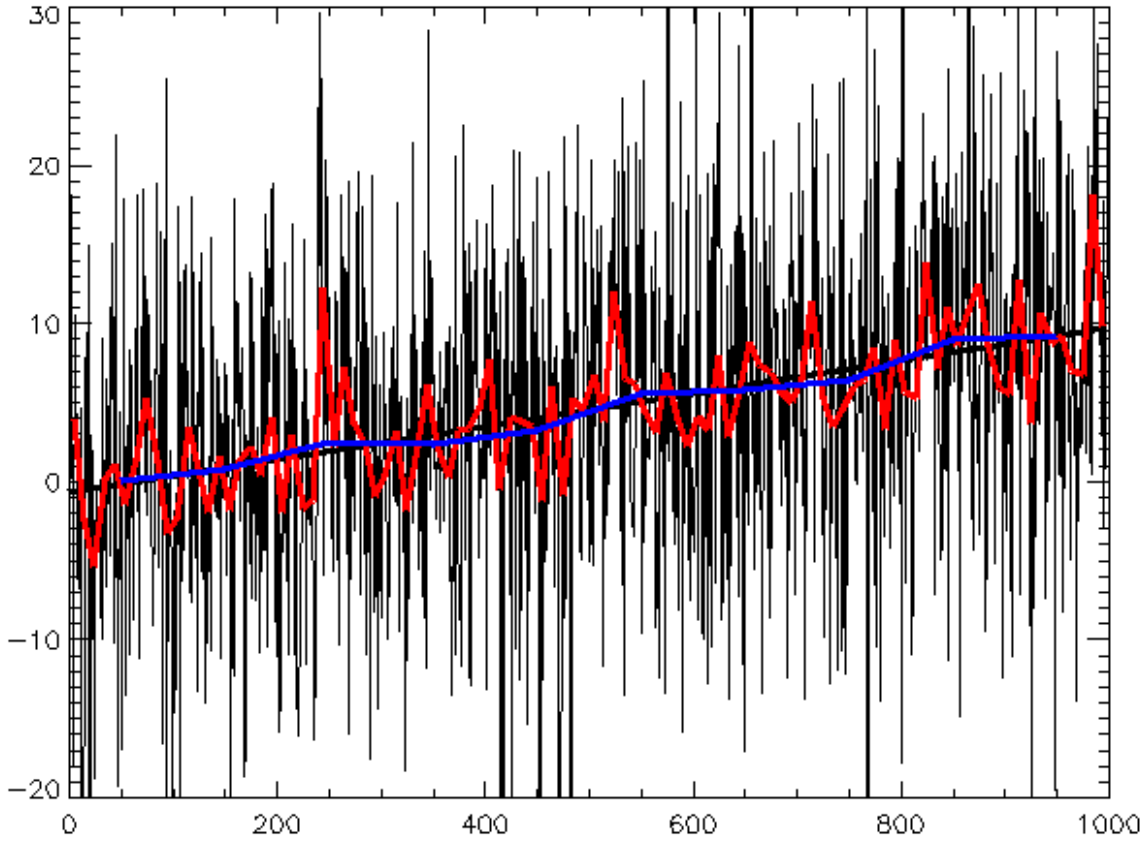
GlobalLogic®




# Time Series: Data Analysis & Forecasting

by Amit Gupta, Senior Consultant
and Akshit Bansal, Associate Consultant

# Contents


Introduction 1

Building Blocks of Time Series Data Analysis 2
- Time Series
- Metrics and Events
- Data Point in Time Series

Characteristics of Time Series 3

Storing Time Series Data 4
- Traditional RDBMS vs Purpose-Built TSDB
- InfluxDB
- TimescaleDB
- Choosing a Time Series Database

Visualizing Time Series Data 6
- Grafana

Time Series and Machine Learning 7
- Training Sets vs Test Sets
- Temporal Dependent Models vs General Additive Models

Summary 8

References 8

# Introduction

Since the dawn of statistical analysis, forecasting has been one of the core pillars of the stream. Analysts have done an incalculable amount of work in creating models from existing datasets to both identify trends and predict how the datasets might look like in future. Whether it is predicting a stock's performance in the market or looking at consumption of a resource by a population, time-based datasets have been the backbone of this field.

The use cases for a time-based data model are seen everywhere in modern statistical analysis. Use varies from something as simple as keeping track of the health of a server cluster, to something as complex as maintaining global weather data for keeping global warming in check. Time is the common dimension which binds all of these data models.

With an increasing dependency on computers for saving, maintaining and analyzing such datasets, a wide array of software solutions have become a new niche in the market. Modern day statistical analysts are not only expected to be a keen observer of patterns; they are also expected to be well-versed in these technologies.

The need for such tools has become such a booming market that even traditional databases are being forced to adapt these concepts. While existing databases are retroactively adding support for storing huge quantities of data only dependent on time, new databases are being built from the ground up to satisfy the needs.

Let's take a look at what exactly is a time-based data series, what it offers and why it is necessary for a modern developer to keep up to date with this tech stack.

# Building Blocks of Time Series Data Analysis

The following terms are used when we talk about time series data analysis.

## Time Series

According to Wikipedia, a time series is a series of data points indexed (or listed or graphed) in time order. Most commonly, a time series is a sequence taken at successive equally spaced points in time. Therefore it is a sequence of discrete-time data. Examples of time series are heights of ocean tides, counts of sunspots, and the daily closing value of the Dow Jones Industrial Average.

## Metrics and Events

A metric is a data point which is gathered on a regular time interval. A common example is a server cluster monitor which tracks disk ops or network usage. These data points are consistent in nature and are most often used for predicting future trends.

An event is a data point which is gathered when something happens. Although they are saved in respect to time, they are not regular and not used for forecasting. A common example would be a logging/tracing system used by software.

## Data Point in Time Series

A time series data point is an immutable record of metrics or events which are always inserted and never updated. Every data point is ideally unique with time being the only relationship between two data points. If you were to plot these data points on a graph, time will always be one of the axes. For example daily temperatures: If you draw a daily temperature graph, time would be on one axis, and the other axis would be temperature.

# Characteristics of Time Series

Time series data is collected at different points in time, as opposed to cross-sectional data which observes individuals, companies, etc. at a single point in time. There is a natural temporal ordering in time series data. What this means is that time series data only cares about the time at which data is pointing at, nothing else.

This is why a study of a tech company's employees' salary on the basis of their experience, for example, is not a time series. In this study, the time at which the salary and experience was stored is irrelevant.

### Seasonality

This is a common characteristic of a time series. It refers to periodic fluctuations in data. Your consumption of ice cream is high every summer and lower in the winter, retail sales increase around holidays and levels out on regular days. This type of variance is very common in a time series.

### Stationarity

This is another characteristic of a time series. While data points might be fluctuating on a shorter scale, over a period of time they remain the same. In purely statistical terms, a time series has constant mean and variance, and covariance is independent of time. Now, of course, not all grey things are elephants and not all time series are stationary. Statisticians use a variety of techniques to convert time series into stationary models.

### Structural Breaks

This refer to sudden unexpected changes in data. This is a common use case in time-series-based data modeling. They help analysts to determine an overall change in the model and where it might lead in the future. A famous example was the changes in macroeconomic indicators from 2008 which foreshadowed the upcoming crash of the U.S. housing market and subsequent recession in global economies.

### Autocorrelation

This is the correlation between observations of the same dataset at different points in time. The need for distinct time series models stems in part from the autocorrelation present in time series data. Imagine a sine wave which has multiple points of high autocorrelation. In simple terms, this means that on a time scale, values will be repeated. An analyst looks at this data and tries to model it in such a way that these repeated patterns can be used for predicting future trends.

# Storing Time Series Data

A common theme which resonates amongst all time series datasets is the sheer number of data points. Imagine this: A fictional company DinosCanFly Inc. created a dinosaur which flies like a helicopter. It is fondly named the DinoCopter.

While testing DinoCopter, the company wanted to store every second of its flight telemetry as a data point. The minimum number of flying hours required for a commercial pilot's license is 1500. That's over 5 million data points just for the flight telemetry; add in other things like parts condition and you are generating data exponentially. Then you realize that all of that is just for a **single** DinoCopter.

Scalability is a major issue for time series databases, which is why it drives the majority of the conversation. The remaining part is performance, of course.

## Traditional RDBMS vs Purpose-Built TSDB

Whether to use a traditional relational database or something purpose-built for storing time series data is a major point of debate among developers.

The arguments for relational databases are simple. They offer high cardinality performance, schema design, vertical scaling and SQL support. But, there are some obvious disadvantages. Relational DBs, and vertical scaling in general, assume that adding resources to a machine is a simple and cheap solution. While this is not entirely wrong, it is not completely right either. There is only so much hardware you can add to a machine.

Another question is the performance and, more specifically, the computational overhead of ACID Time series data that tries to look at trends over a period of time rather than individual data points. Durability of data is not that necessary when compared to more traditional scenarios.

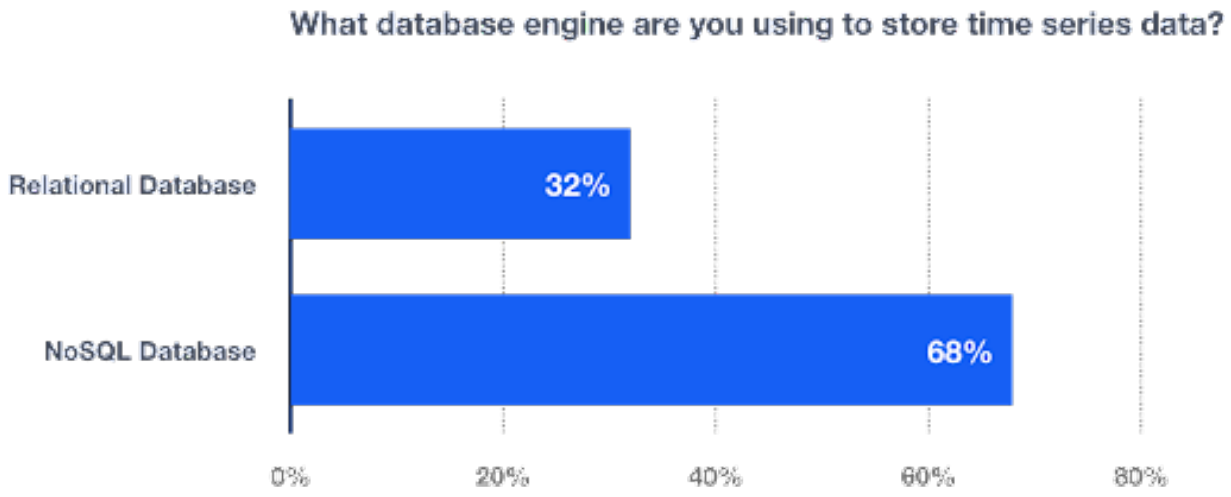
Purpose-built TSDBs solve some of these problems. They are mostly NoSQL-based DBs which offer BASE, high horizontal scalability, and low cardinality performance.

BASE sacrifices some features of ACID to make sure that data is always available. The assumption is: losing some data points is not that big of a loss in the grand scheme of things. Since they are not conforming to ACID, horizontal scalability is almost an out of box feature. Since they are NoSQL there is no schema overhead with time being the only defining feature. The associated data uses rudimentary types to store values with no close association between two points other than time. This, in turn, offers great low cardinality performance.

Some performance benchmarks show that in such DBs low cardinality data can be written at considerably higher speeds.

But there are disadvantages to such solutions, as well. First and foremost is the loss of SQL support. These systems ship with their own query languages which have their own nuances. This means an increased learning curve and unknown idiosyncrasies. Reliability is another issue. Relational DBs have been around for decades with great support and reliability. In contrast, these new DBs are still evolving and maturing. This makes them a risk for enterprise solutions.

Following is the finding of one of the surveys done by Percona:




## InfluxDB

InfluxDB is an open source TSDB purpose built for time series data. It uses a NoSQL approach for storing data and is the market leader in this field.

It is written in GO and is optimized for fast, high-availability time series data storage and retrieval. It offers its own query languages, an "SQL like" InfluxQL and a more custom NoSQL, Flux. Both languages are powerful tools which offer a variety of features.

InfluxDB is a part of the TICK (Telegraf InfluxDB Chronograf Kapacitor) stack. This stack offers open source tools for writing, storing, processing and visualizing time series data. While InfluxData, the parent company of InfluxDB, offers this entire stack, InfluxDB is also compatible with other stacks, the most popular combination being InfluxDB with the visualization tool Grafana.

### TimescaleDB

While InfluxDB is the standard bearer of NoSQL approach, TimescaleDB is on the other end of the spectrum. It is an open source TSDB built on top of PostgreSQL and is gaining a lot of traction in the market.

The reason behind Timescale's success is simple. It offers all the features of a relational SQL DB while trying to mitigate the disadvantages. It promises scalability and high cardinality performance.

Since the DB uses an SQL approach, there is no learning curve in querying the data. All the while, it comes in with the inherent stability and the proven mettle of a PostgreSQL platform.

### Choosing a Time Series Database

Like most things in development and architectural design, the decision to pick a proper TSDB is based on choosing your tradeoffs. What does your system need more of, high availability or high durability? Do you value scaling over performance? The best way to reach a conclusion would be to clearly analyze your requirements and make an informed decision.

## Visualizing Time Series Data

While most analysts are right in assuming that there is no better way to analyze time series data than creating robust and testable models, just being able to view the data in a visual manner is actually really helpful.

Complex models which require a lot of calculation and analysis to provide reasonable trends and conclusions are certainly an end goal, but not all problems are created equal. Keeping that in mind, a number of visualization tools for time series data have come to the market. These tools offer a lot of out of the box features like the ability to transform data into human- readable graphs, charts and panels. They offer support for multiple data sources and optimized query generation.

### Grafana

Grafana is a popular open source visualization and analytical suite mainly used for time series data. It provides ways to create, explore, and share time series data in easy-to-understand graphical representations.

Grafana offers a plethora of visualization options and customizations. These help to create informative and smart dashboards where any time series can be visually represented. Further, there is a vast library of plugins and dashboards which can cater to more specific requirements. It supports over a dozen TSDBs natively and the installation is hassle-free. The UI is intuitive and very robust. It also offers modern features like LDAP support for more enterprise oriented use cases.

# Time Series and Machine Learning

Classical statistical analysis has developed many techniques like Exponential Smoothing and Fourier Transform to model time series data. Data scientists are now working to use this data and ML to create time series models that are both accurate in their forecast and allow for the inclusion of influential features on the time series.

To model time series data using machine learning, the time feature must be broken out into subcomponents of the time series. The subcomponents can include durations as small as a day or as large as a quarter. The important thing is, each one of those broken time features should be able to capture some part of the actual time series data. Using this approach, it becomes possible to gain valuable insight into the time features that may influence our data, for example: a time series that tracks sales for a retail store. By breaking the data into smaller features it may be possible to predict sales for upcoming months.

## Training Sets vs Test Sets

Like in all ML models and processes, a training and test set of data is necessary to be created. The core idea is to use training data to build a new ML model and then use that model against test data to test the model's abilities.

A common way to achieve this is finding a cutoff point in the time series. The data prior to the cutoff point is used as training data for creating the model. The data after the cutoff point is used as a test set. The thing to keep in mind here is that the order of the time series should be maintained after the data is split.

Another approach is called "windowing." The core idea here is that a small range of dates are essentially used as a "window" for training the model. The immediate next "window" is used for testing the model. This way, the window keeps sliding forward and the order of time series is maintained. This is similar to any cross-validation but the subsets aren't random.

## Temporal Dependent Models vs General Additive Models

TDMs are the intuitive way of looking at time series and making forecasts. Let's say you want to predict stock prices. It makes more sense to look at yesterday's values to predict tomorrow's values, rather than last year's. These correlations between past and present values demonstrate temporal dependence, which forms the basis of a popular time series analysis technique called ARIMA (Autoregressive Integrated Moving Average).

However, ARIMA makes rigid assumptions. To use ARIMA, trends should have regular periods, as well as constant mean and variance. In simpler terms, we have to first apply a transformation to the trend so that it is no longer increasing but stationary. Moreover, ARIMA cannot work if we have missing data.

GAMS don't sum effects of individual predictors, but rather they are a sum of smooth functions. Functions allow us to model more complex patterns, and they can be averaged to obtain smoothed curves that are more generalizable.

Because GAMs are based on functions rather than variables, they are not restricted by the linearity assumption in regression that requires predictor and outcome variables to move in a straight line.

The general idea is that machine learning, while not always the perfect choice, can be powerful in modeling time series data due to its ability to handle non-linear data. It's also worth noting that having more training data in a time series need not necessarily lead to more accurate models. Anomalous values or rapidly changing trends could upend any prediction efforts. Worse still, sudden shocks that permanently affect a time series could also render all past data as irrelevant.

Therefore, time series analysis works best for trends that are steady and systematic, for which we can assess with visualizations.

# Summary

Time series data and related technologies are here to stay. While statistical analysis has always been used to identify patterns and trends, time series technologies have discovered new avenues of growth. Using technology and tools, which are largely open source, analysts can work on something as menial as predicting server outages under upcoming events to predicting the GDP growth of a budding economy.

Using time series data and ML has also provided a new outlook to data scientists. While not a perfect match, time series and ML do go hand in hand in creating models which can solve a lot of complex problems.

What really remains to be seen is how this technology will mature. When the NoSQL movement first started, people really jumped to conclusions in predicting the death of traditional DBs. But like anything related to technology, nothing is ever truly dead. SQL seems to be making a comeback with more features and improvements. The same can happen to TSDBs. Right now they look like a perfect solution for our problems but maybe we need to read the time series data to predict their future.

# References


Open Data Science

Influx Data: Time Series Database

What the Heck is Time Series Data and Why Do I Need a Time Series Database?

## About the Authors

Amit Gupta is Sr. Consultant at GlobalLogic. He has vast experience in product transformation from on-premise to cloud.

Akshit Bansal is Associate Consultant at Globallogic. He is an expert in Analytics and Data Security.

GlobalLogic®

GlobalLogic, a Hitachi Group Company, is a leader in digital product engineering. We help our clients design and build innovative products, platforms, and digital experiences for the modern world. By integrating our strategic design, complex engineering, and vertical industry expertise with Hitachi's Operating Technology and Information Technology capabilities, we help our clients imagine what's possible and accelerate their transition into tomorrow's digital businesses. Headquartered in Silicon Valley, GlobalLogic operates design studios and engineering centers around the world, extending our deep expertise to customers in the automotive, communications, financial services, healthcare & life sciences, media and entertainment, manufacturing, semiconductor, and technology industries.



www.globallogic.com