O'REILLY®

PREVIEW EDITION

Practical Time Series Analysis

PREDICTION WITH STATISTICS AND MACHINE LEARNING

Aileen Nielsen

Practical Time Series Analysis

Prediction with Statistics and Machine Learning

This Preview Edition of *Practical Time Series Analysis*, Chapters 1 and 4, is a work in progress. The final book is currently scheduled for release in November 2019 and will be available at oreilly.com and other retailers once it is published.

Aileen Nielsen

Beijing • Boston • Farnham • Sebastopol • Tokyo

O'REILLY®

Practical Time Series Analysis
by Aileen Nielsen

Printed in the United States of America.

Published by O'Reilly Media, Inc. , 1005 Gravenstein Highway North, Sebastopol, CA 95472.

O'Reilly books may be purchased for educational, business, or sales promotional use. Online editions are also available for most titles (*http://oreilly.com/safari*). For more information, contact our corporate/institutional sales department: 800-998-9938 or corporate@oreilly.com .

Editors: Jonathan Hassell and Jeff Bleiel
Production Editor: Katherine Tozer
Interior Designer: David Futato
Cover Designer: Karen Montgomery
Illustrator: Rebecca Demarest

November 2019: First Edition

Revision History for the First Edition
2019-01-18: First Release

See *http://oreilly.com/catalog/errata.csp?isbn=9781492041658* for release details.

978-1-492-04165-8

[LSI]

Table of Contents

Foreword

The opportunity for time series data is enormous. Both existing transaction records and new Internet of Things (IoT) information have tremendous potential for generating actionable insights and improving organizational performance. Changes in market conditions, customer interactions, machine behavior, and fraudulent activities can be tracked and responded to using time series data. The value of time series data means that anyone currently architecting, building, or deploying data-driven applications or analytics should understand how to best work with this data format.

In *Practical Time Series Analysis: Prediction with Statistics and Machine Learning*, author Aileen Nielsen describes how to work with time series data for the purposes of insights. There are practical context on how statistics and machine learning use time series data along with how time series are used in various types of applications. She also describes the variety of technology solutions used for managing time series data (including the differences between NoSQL and SQL approaches), working with live vs historical data, flat file solutions, and more.In this free excerpt, we feature two key chapters from *Practical Time Series Analysis*:

A Historical Perspective of Time Series Data (Chapter 1)
: When time series analysis became interesting and how statistics and machine learning have followed the data.

Storing Temporal Data (Chapter 4)
: Describes the requirements for analyzing time series data, comparing and contrasting SQL vs NoSQL solutions along with popular database solutions

MemSQL is proud to offer you this free excerpt, as we believe that every business should be able to easily unlock the full potential of their data. Current databases are not scalable for most time series applications. With MemSQL, you have the ability to ingest all the time series data you receive and leverage it for real-time decision making and for use by applications at will. With MemSQL, you can simultaneously make time series data available for fast analytics, machine learning and AI, and to as many

users as you like, all within a familiar SQL interface. This unmatched capacity applies equally to time series data, JSON, geospatial, full text, and traditional relational data sets.

We hope you enjoy this excerpt from *Practical Time Series Analysis*. We encourage you to put the information to use in your own applications.

— Peter Guagenti
MemSQL
January 2019

CHAPTER 1

Time Series: A History and a Future Forecast

Welcome to practical time series analysis. This book is intended for both veteran cross-sectional data analysts looking to expand their interests and for new data analysts seeking an overview of an increasingly relevant way of analyzing data. Unfortunately, time series analysis has often been overlooked in the standard data science tool kit, but that is starting to change as more and more panel data is collected from a wide range of scenarios, including IoT, big data in healthcare, and smart cities.

As continuous monitoring and data collection grows increasingly common, the need for competent time series analysis with both traditional statistical methods and innovative machine learning techniques will continue to grow. This book will provide you with a background in a broad range of time series analysis techniques useful for the kinds of data most useful to understand - and predict - human behavior, scientific phenomena, and industrial pipelines.

Let's start from scratch. What is time series analysis? Broadly speaking, time series analysis is the endeavor of extracting meaningful summary and statistical information from points arranged in temporal order. This is done to diagnose past behavior as well as to make predictions about likely future behavior.

The most common concerns of time series analysis are forecasting the future and classifying the past. In this book we will address both such analyses with a variety of approaches, ranging from traditional statistical models developed a century ago to the cutting edge deep learning architectures published on arXiv only a few months prior to this book's publication. None of these techniques have developed in a vacuum or out of purely theoretical interest. Every time series technique there ever was grew out of new ways of collecting, recording, and visualizing data, and that connection is what we will think about in this chapter.

Time Series in Diverse Applications

Time series analysis includes questions of causality: how did the past influence the future? Often such problems, and their solutions, go unlabeled as time series problems. That's generally a good thing, as it means thinkers from a variety of disciplines have contributed novel ways of thinking about time series data sets while often rediscovering some of the most important techniques in a variety of different disciplinary settings and data sets.

Here we briefly survey few disciplines where time series is a core issue, even if not always recognized as such.

Medicine as a time series problem

Medicine got a surprisingly slow start to thinking about the mathematics of predicting the future, despite the fact that getting as accurate a prognosis as possible would always have been important to patients. This was the case for many reasons. Mathematics, as we know it, is a recent phenomenon, particularly with respect to statistics and probabilistic processes. Also, most doctors practiced in isolation, without easy professional communications and without formal record-keeping infrastructure for patient or population health measures. There were no mechanisms in place to accrue data, so physicians took each case on its own merits or with very small N as observations in their own lifetimes.

So it is not too surprising that one of the early mathematical innovations related to medicine came not from a physician but from someone who might have been more accustomed to recordkeeping given the demands of a merchant's profession. Such an innovation came from John Graunt, a 17th century London haberdasher.

Graunt is considered to be one of the originators of the discipline of demography. He was the first creator of a life table. This may be more familiar to you as what are commonly referred to as actuarial tables. These give the probability that a person of a given age will die before her next birthday, and Graunt was the first person in the West known to have formulated and published these life tables, in the process becoming one of the first people to apply statistics to a question related to medicine and human health.

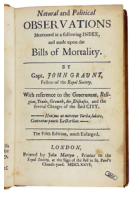

Figure 1-1. John Graunt's actuarial tables were one of the first results of time series style thinking applied to medical questions. Image is in the public domain and taken from the Wikipedia article on John Graunt.

Unfortunately, Graunt's way of thinking mathematically about human survival did not "take." Even after a sensible human[1] realized that deaths worked somewhat systematically, medicine largely missed out on two centuries of taking advantage of this insight. Medical publications continued to be mechanism oriented with a focus on a small N (or single N) cases. These publications tended to lack time series specific language or concepts even when descriptions of patient health trajectories were included in illness descriptions.

Even as the modern world as we know it today began to develop - complete with professional societies, scientific journals, and government-mandated health record keeping, the focus in medicine continued to be on mechanisms rather than statistics and time series. Again, there were quite understandable reasons for this, even trade-offif it seems unfortunate in retrospect. First, the study of anatomy and physiology in small numbers of subjects had provided the major advances in medicine for centuries, and most humans, even scientists, tend to keep doing what works for as long as possible. While a focus on physiology was so successful, there was no reason to look further afield.

Second, there was very little reporting infrastructure in place for physicians to tabulate and share information on the scale that would make time series methods superior to clinical observations stored as cross-sectional data, given the constraints on physicians' ability to gather reliable data. Time series is the most demanding statistical endeavor with respect to record keeping because records must be linked together over time. Time series as an epidemiological practice has only emerged very late and slowly, once there was sufficient governmental and scientific infrastructure in place to ensure reasonably good and lengthy temporal records. Indeed, healthcare as time series analysis remains a young and challenging field, and it remains quite difficult to forecast health related outcomes, such as the course of an epidemic or even the length of the annual flu season. Even for small case-study based research, maintaining con-

1 A sensible human who even had the wherewithal to publish a book pointing this out!

tact with the same group of individuals and maintaining their participation in a research cohort is excruciatingly difficult and extraordinarily expensive. When such studies are maintain for long periods of time, they tend to become extremely famous - and repeatedly, or even excessively researched - because they are so difficult to obtain in the first place[2].

Interestingly, time series analysis for individual patients has a far earlier and more successful history than do population-wide studies. Explicit time series analysis made its way into medicine even before the standardization of the randomized control study over a century ago, when electrocardiograms[3] (ECGs) were invented in 1903. Another time series machine, the electroencephalogram[4] (EEG) was introduced into human medicine in 1924. Both of these time series tools were part of a larger wave of the enhancing medicine with measuring instruments, as physicians re-purposed ideas and technologies coming out of the Second Industrial Revolution and put them to use with the human body.

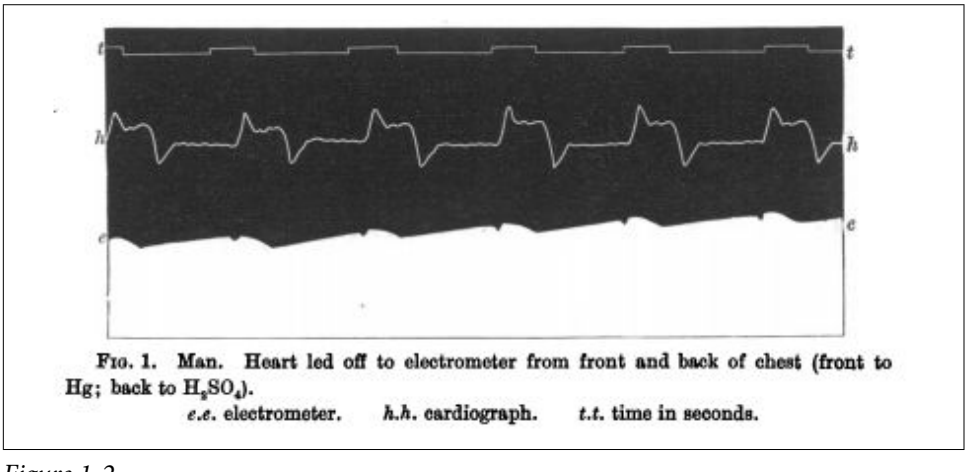

FIG. 1. Man. Heart led off to electrometer from front and back of chest (front to Hg; back to H_2SO_4).
e.e. electrometer. *h.h.* cardiograph. *t.t.* time in seconds.

Figure 1-2.
An early ECG recording from the original 1877 paper. Image is in the public domain and taken from A Demonstration on Man of Electromotive Changes Accompanying the Heart's Beat by A. D. Waller, M.D.

2 In 2019 and beyond, we happen to be in luck, as this problem is rapidly going away. More on this soon.

3 ECGs record the electrical signals passing through the heart, which can be used to diagnose a variety of heart conditions.

4 EEGs measure electrical impulses in the brain and are able to do so non-invasively.

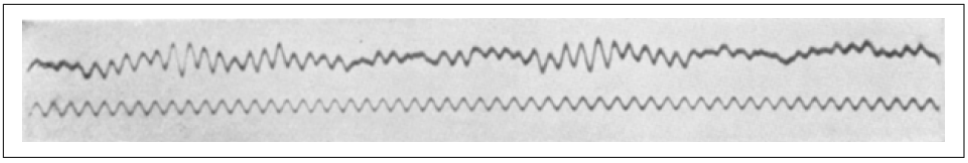

Figure 1-3.
The first human EEG recording, from 1924. Image is in the public domain and taken from the Wikipedia article on EEGs.

As you will see as you begin to search the internet for health-related time series data, ECG and EEG time series classification and forecasting remains an active and interesting area of research for very practical purposes, such as estimating the risk of a sudden cardiac crisis or predicting a seizure. These are a rich source of very interesting data, but one "problem" with such data is that it tends to only be available for patients having specific problems in a specific area of the body. These machines do not make available multi-variable long-range time series that can tell us more broadly about human health and behavior.

We are lucky, however, that we are well past an era where ECGs and EEGs are the dominant medical time series available. Nowadays, healthcare is benefiting from the digital revolution of the late 20th/early 21st century. With the advent of wearable sensors and smart electronic medical devices, we are entering an era where healthy adults are taking routine measurements, either automatically or with minimal manual input. We are on the brink of having a tremendous trove of new datasets, both about sick people and healthy people alike. This is in stark contrast to the last century's worth of medical time series data, which was almost exclusively measured on sick people and in which medical time series data was expensive and extremely difficult to access.

As recent news coverage has shown (such as here and here), a variety of non traditional players are attempting to get into the medical field, ranging from enormous social media companies to financial institutions to retail giants. They likely all plan to use large data sets from a variety of sources to gain novel insights into healthcare. It's clear that - thanks to burgeoning modern healthcare data sets - both healthcare and time series analysis will rapidly evolve in the coming years.

Forecasting economic growth

As Harvard Business School professor Walter A. Friedman tells it, economic forecasting grew out of the anxiety triggered by episodic banking crises in the United States and Europe in the late 19th and early 20th centuries. Entrepreneurs and researchers alike drew inspiration from the idea that the economic system could be likened to a cyclical system just like the weather was thought to behave.

With the right measurements, it was thought, predictions could be made and crisis averted. Even the language of early economic forecasting mirrored the language of weather forecasting. This was unintentionally apt. In the early 20th century, both economic and weather forecasting were indeed alike - both pretty terrible at predicting the future. But the economists' aspirations, though slow to achieve and unrealistic for their time, created an environment in which progress could at least be hoped for and some meaningful concrete efforts established.

Early economic forecasting efforts led to the creation of economic indicators (and tabulated, publicly available histories of those economic indicators) that are still in use today. Nowadays, the United States and most other highly technological nations have thousands of government researchers and record keepers whose jobs are to record data as accurately as possible and to make this data available to the public as easily as possible. Such record keeping by governments around the world has proven invaluable to economic growth as well as to transportation providers, manufacturers, small business owners, and even farmers.

BUSINESS CYCLE REFERENCE DATES		DURATION IN MONTHS			
Peak	Trough	Contraction	Expansion	Cycle	
Quarterly dates are in parentheses		*Peak to Trough*	*Previous trough to this peak*	*Trough from Previous Trough*	*Peak from Previous Peak*
	December 1854 (IV)	--	--	--	--
June 1857(II)	December 1858 (IV)	18	30	48	--
October 1860(III)	June 1861 (III)	8	22	30	40
April 1865(I)	December 1867 (I)	32	46	78	54
June 1869(II)	December 1870 (IV)	18	18	36	50
October 1873(III)	March 1879 (I)	65	34	99	52
March 1882(I)	May 1885 (II)	38	36	74	101
March 1887(II)	April 1888 (I)	13	22	35	60
July 1890(III)	May 1891 (II)	10	27	37	40
January 1893(I)	June 1894 (II)	17	20	37	30
December 1895(IV)	June 1897 (II)	18	18	36	35
June 1899(III)	December 1900 (IV)	18	24	42	42
September 1902(IV)	August 1904 (III)	23	21	44	39

Figure 1-4. The US federal government funds many government agencies and related non-profits who record vital statistics as well as formulating economic indicators. This screenshot is taken from a table of statistics about business cycles provided by the National Bureau of Economic Research.

Much of this data, particularly the most newsworthy, tends to be about economic data that proxies for the overall well-being of the population and size of the population's economic activity for a given year. Such vital information comes particularly from the

number of unemployment benefits requests, estimates of the gross domestic product, and information about tax returns.

Thanks to the desire for economic forecasting, the government became a curator of data as well as a collector of taxes. Reciprocally, it was the collection of this data that enable modern economics, the modern finance industry, and data science generally to blossom. More importantly, we now safely avert many more banking and financial crises than any government was able to avert in past centuries.

Let's get back to the historical side of things. Private organizations began to copy government record-keeping, inspired by the obvious success of tabulating and publicizing such numbers. Over time commodities and stock exchanges became increasingly technical and increasingly well documented and formal. Financial almanacs became popular, too. This happened both because market participants became more sophisticated and because emerging technologies enabled greater sophistication and new ways of competing and thinking about prices.

All this minute record-keeping inevitably gave rise to the pursuit of making money off the markets via math rather than intuition, in a way driven entirely by statistics (and later by machine learning). Early pioneers did this by hand, whereas current "quants" do this by very complicated and proprietary time series analytic methods.

One of the pioneers of 'mechanical trading', which can be seen as time series forecasting via algorithm, was Richard Dennis. Dennis was a self-made millionaire who famously turned ordinary people, called the Turtles, into star traders by teaching them a few select rules about how and when to trade. These rules were developed in the 1970s and 1980s and mirrored the "AI" thinking of the 1980s, in which heuristics still strongly ruled the paradigm of how to build intelligent machines to work in the real world.

Since then many 'mechanical' traders have adapted these rules, and for that reason the rules have become less profitable in a crowded automated market with many copy cats. 'Mechanical' traders continue to grow in numbers and wealth, but because their domain os competitive, they are continually in search of the next best thing, creating many new kinds of time series analytical techniques that empirically work, although sometimes with little theoretical justification. The disconnect between what works and what can be theoretically justified can be stark in time series, and this is particularly true in the financial markets and for related economic indicators.

Forecasting weather

For obvious reasons, predicting the weather has long been a preoccupation and passion of humanity. This problem has even interested some of the most famous philosophers. Aristotle delved into weather with an entire treatise on the subject, and his ideas about the causes and sequencing of the weather remained common currency

until the Renaissance, when observers realized that a more data-driven approach was required.

From the Renaissance onward, scientists around Europe began building instruments to record the state of the atmosphere, and in doing so they recorded time series, at daily or even hourly intervals. These records were kept in a variety of locations, from private diaries for individual reference to town and national governmental records. For centuries this remained the only way that Western civilization tracked the weather - scattershot informal diary entries, or sometimes something a little more systematic.

It was only in the late 19th century - hundreds of years after many basic weather-related instruments had come into use - that the telegraph allowed compilations of atmospheric conditions in parallel time series depending on location. This became standard by the 1870s and led to the creation of the first meaningful data sets for making local weather predictions based on what was happening in many locations. Such a development was important because - as we will see in various data analyses in this book - predicting the future is easier when you have many parallel time series containing different kinds of information rather than just a single history of a single metric.

By the turn of the 20th century, the idea of predicting the weather with numerical methods was vigorously pursued with the help of these newly compiled data sets. Such early efforts represented a spectacular amount of effort but very poor results. Physicists and chemists had well proven ideas about the relevant natural laws, but there were too many natural laws and measurements to take into account. The resulting system of equations was so complex that it was a notable scientific breakthrough the first time someone even attempted to do the calculation. He spent weeks completing the deed, and his prediction was entirely wrong - but at least he had attempted it.

Several decades of research followed before it was understood how to simplify the physical equations in a way that increased accuracy and computational efficiency. This tradition has been handed down even to current weather prediction models, which operate on a mix of known physical principles, computational shortcuts, and proven heuristics. Nowadays many governments make highly granular weather measurements from hundreds or even thousands of weather stations around the world, and these predictions are grounded in data with precise information about weather station locations and equipment. The roots of all such efforts trace back to the coordinated data sets of the 1870s.

Predicting the weather remains fundamentally a time series problem where past measures are used to predict future ones, both with physical reasoning and also with statistical assumptions. In recent years, methods with more statistical grounding are increasingly overtaking those with physical grounding, and one day an AI-based program with deep learning on a plethora of data may do better than a Ph.D. scientist in

possession of all the physics that should predict the system (though it's unclear whether such a possibility would be cause for celebration if the AI program couldn't 'explain' how it had made its predictions).

Recent startup efforts to apply artificial intelligence to weather forecasting will help farmers and retailers alike understand how the weather will affect business. These statistical and machine learning methods need not exclude the physics oriented methods but can eventually pair with them.

Astronomy

Astronomers are masters of time series, both for calibrating instruments and for studying their objects of interest as those objects change position or quality of their emissions over time. Astronomy has traditionally relied heavily on plotting objects, trajectories, and measurements over time. For example, sunspot time series were recorded in ancient China as early as 800 B.C.. Indeed, an enormous trove of sunspot data makes this one of the most well recorded natural phenomena throughout history.

Some of the most exciting astronomy of the 20th and 21st century relates to time series analysis. In particular, the discovery of variable stars and the observation of transitory events, such as supernovae, are the result of constantly monitoring live streams of time series data based on the wavelengths and intensities of light. Time series have had a fundamental impact on what we can know and measure about the universe.

Incidentally, this constant monitoring has even allowed astronomers to catch events as they are happening (or rather as we are able to observe them once the light depicting the event has traveled for years or even millions of years to reach us). These breakthroughs have led to better understandings of the nature of stars as well as better ways to measure celestial distances.

In the last few decades, the availability of explicitly time stamped data, as formal time series, has exploded in astronomy with a wide array of new kinds of telescopes collecting all sorts of celestial data. Some astronomers have referred to a time series "data deluge" resulting from large scale photometric surveys. There are tremendous opportunities for additional discoveries in astronomy via time series reasoning.

When Time Series Analysis Became Interesting

The Box-Jenkins model is a standard approach to choosing parameters for an autoregressive moving average process to model a time series on the basis of its past values and to use that model to predict the future.

Are you already losing interest? Those words, autoregressive...moving….average…. will flow off your tongue in a few chapters, but it's understandable that your eyes glaze over right now.

"Autoregressive moving average" sounds rigorous and scientific, but in practice the process of predicting the future, or seeking to explain the past, is notoriously controversial and messy. This is almost as true in statistics and machine learning as it is in history, politics, or sociology. As with those disciplines, any data set showing changes over time invariably has too much noise and too few parameters to explain the world as neatly and tightly as we would like.

George Box, one of the statisticians who originated the Box-Jenkins model appreciated this as much as anyone. As he explained, **All models are wrong, but some are useful.**

The relevant question about a model should always be one of its utility rather than its correctness, and this is especially true when it comes to the slightly dubious endeavor of trying to predict the future.

Box made his famous statement in response to a common attitude, prevalent among some statisticians at that time, that proper time series modeling was a matter of finding the best model to fit the data. As Box explained, the idea that a mere model that you can write down (or code up) can truly describe the real world in all its complexity is very unlikely. Box made this pronouncement in 1978, which seems bizarrely late into the history of as important a field as time series analysis, but the field itself is indeed surprisingly young.

The Box-Jenkins model itself only appeared in 1970, and it appeared first not in an academic journal but rather in a statistics textbook, *Time series analysis: Forecasting and control*. Incidentally this textbook remains popular and is now on its 5th edition. The original Box-Jenkins model was applied to a data set of carbond dioxide levels emitted from a gas furnace. While there is nothing quaint about a gas furnace, what does feel quaint by today's standards is the size of the data set, coming in at fewer than 300 data points.

The modern data analyst would be somewhat nonplussed if asked to analyze such a small data set to come up with predictions about the future. However, she could be comforted in knowing that the Box-Jenkins model and other linear methods have stood up pretty well against modern methods applied on larger data sets. It's quite likely that larger data sets were available in the 1970s, but remember that large data sets might have been more of a pain than was really worth it in those days. We are talking about an era that predates not only conveniences such as R and Python but even "heavy duty" analytical tools such as C++. Compared to what was available then, even the slowest and worst setup today would have been a joy. Researchers had good

reasons to look for methods that worked for small data sets, preferably with as little computational iteration as possible.

Due to the technical limitations and tedium, time series analysis and forecasting developed as computers did, with larger data sets and easier coding tools paving the way for more experimentation and more interesting questions. Professor Robert Hyndman's history of forecasting competitions provides apt examples of how time series forecasting competitions developed at a rate parallel to that of computers.

Professor Hyndman puts the "earliest non-trivial study of time series forecast accuracy" as 1969 in a doctoral dissertation at the University of Nottingham (just a year before the publication of the Box-Jenkins method). That first effort was soon followed by organized time series forecasting competitions, the earliest ones featuring data sets of around 100 in size in the early 1970s. Not bad, but surely something that could be done by hand if absolutely necessary.

By the end of the 1970s, researchers had put together a competition with around 1000 data sets, an impressive scaling up. Incidentally, this time was also marked by the first commercial microprocessor, the development of floppy disks, Apple's early personal computers, and the computer language Pascal.

Surely some of these innovations had something to do with the forecasting competition scaling up so much. In fact, this was a decade of particularly important growth with the introduction of smaller personal computers. A time series forecasting competition of the late 1990s included 3000 data sets, a paltry scaling up in two decades compared to the progress in the 1970s alone. Nothing important was happening in computing during those decades, and the progress looks quite dull compared to what we have seen in the last 20 years as wearables, machine learning and GPUs have revolutionized the amount and quality of data available for study.[5] While these collections of data sets were substantial and no doubt reflected tremendous amounts of work and ingenuity to collect and curate, they are dwarfed by the amount of data now available, even the data for just one person. Similarly, it would likely be possible in the space of a few days for an intermediate Python programmer to take advantage of governmental open data initiatives to scrape thousands of time series in the space of a week or two. **Time series data is everywhere, and soon everything will be a time series.**

5 Given the array of gadgets humans can carry around with them as well as the timestamps they create in their environment as they shop for groceries, log into a computer portal at work, browse the internet, check a health indicator, make a phone call, or drive in traffic recorded by GPS systems circling the earth, we can safely say that an average American likely produces thousands of time series data points every year of his life.

The Origins of Statistical Time Series Analysis

Statistics is a very young science relative to similarly popular and useful technical fields. **The history of statistics, data analysis, and time series has always depended strongly on when, where, and how data was available and in what quantity.** For this reason, the ability of time series analysis to exist as a discipline is linked to developments in probability theory but equally to the development of stable nation states, where keeping records first came to be a realizable and interesting target. Given that context, much early time series work related to questions of promoting the social good, from predicting who would die to what the weather would be like and when a bank would next fail. However, science offered opportunities aplenty as well.

One benchmark for the beginning of time series as a discipline is the application of autoregressive models to real data. This didn't happen until the 1920s with the application of such methods to, among the earliest data sets, sunspot models (which will be discussed later in this chapter). Udny Yule, an experimental physicist turned statistical lecturer at Cambridge University, applied an autoregressive model to sunspot data, offering a novel way to think about the data in contrast with the spectral techniques that had been used earlier on the data to assess the periodicity. Yule pointed out that an autoregressive model did not begin with a model that assumed and almost enforced periodicity.

When periodogram analysis is applied to data respecting any physical phenomenon in the expectation of eliciting one or more true periodicities, there is usually, as it seems to me, a tendency to start from the initial hypothesis that the periodicity or periodicities are masked solely by such more or less random superposed fluctuations -- fluctuations which do not in any way disturb the steady course of the underlying periodic function or functions...there seems no reason for assuming it to be the hypothesis most likely a priori.

Yule's thinking was of course his own, but we can imagine some social influences leading him to notice how a model presupposed its own outcome to some extent and wanting to break from that. As a former experimental physicist who had worked abroad in a physics laboratory in Germany, Yule would certainly have been aware of the recent rapid recent in quantum mechanics, itself a probabilistic field and one that represented a fundamental and ground-breaking paradigm shift from earlier physics.

Contemporaneously, statisticians of the 1920s were largely interested in designing experiments properly and wrote widely on how scientific researchers should deploy statistical methods. As physics shifted its paradigm, statistics sought to find truth without making assumption or building in conclusions that would prevent similar such paradigm shifts when the scientific data required. Yule's paper reflected that process of a fundamental disciplinary shift and offered a broader view of how to analyze time series data than had been present before. Yule's paper can be seen as a natural

outgrowth of the contemporaneous scientific and statistical challenges and victories he saw happening around him.

But science wasn't the main attraction to interest in time series. As the world became a more orderly, recorded, and predictable place, particularly after World War II, early problems in practical time series analysis were primarily presented by the business sector. Business-oriented time series problems were important and not overly theoretical in their origins. These included forecasting demand, estimating future raw materials prices, and beginning to develop hedging techniques to stabilize manufacturing costs.

Techniques were adopted when they worked and rejected when they didn't. It probably helped that industrial workers had access to larger data sets than were likely publicly available to academics at the time (as continues to be the case, a particularly important problem for deep learning). This meant that sometimes practical but theoretically under-explored techniques came into widespread use before they were well understood. For example, exponential smoothing, was already widely used long before academics understood why exponential smoothing works so well. In this way, time series mirrored the broader history statistics itself, where industry provided many important contributions inspired by practical needs (e.g. Student's t-test).

Despite decades of time series analysis and forecasting in industry, there was no proper textbook for time series until the 1970s. However, the success story of statistical time series analysis is the success story of technology. As more data became available, as well as more computing power, statisticians were able to explore more complex models for time series.

The Origins of Machine Learning Time Series Analysis

Early machine learning in time series forecasting dates back as far as does proper statistics. An oft-cited paper from 1969, "The Combination of Forecasts", analyzed the idea of combining forecasts rather than choosing a "best" one as a way to improve forecast performance. This idea was, at first, abhorrent to traditional statisticians, but ensemble methods have come to be the gold standards in many forecasting problems. Such ensembling rejects the idea of a perfect or even significantly superior model relative to all possible models.

More recently, ractical uses for time series analysis and machine learning emerged as early as the 1980s, and included a wide variety of scenarios.

- Computer security specialists proposed anomaly detection as a method of developing intrusion detection systems.
- Dynamic time warping, one of the dominant methods for "measuring" the similarity of time series, came into use in the 1980s, when computing power would

finally allow reasonably fast computation of "distances" say between different audio recordings.

- Recursive neural networks were invented and shown to be useful for recovering stored patterns from corrupted transmitted versions.

Time series analysis and forecasting have yet to reach their golden period. Unlike, image analysis, time series analysis remains dominated by traditional statistical methods as well as simpler machine learning techniques, such as ensembles of trees and linear fits. We are still waiting for a great leap forward for predicting the future, particularly with respect to deep learning.

More Resources

History of Time Series Analysis and Forecasting

Revisiting Francis Galton's Forecasting Competition
K. Wallis

This tour through a very early example of averaging predictions to make a forecast is a historical and statistical tour of a very early paper on forecasting the weight of a butchered ox while the animal was still alive at a county fair.

On a Method of Investigating Periodicities in Disturbed Series, with special reference to Wolfer's Sunspot Numbers
G.U. Yule

Ufny Yule's seminal paper, and one of the first applications of autoregressive moving average analysis to real data, also illustrating a way to remove the assumption of periodicity from analysis of a putatively periodic phenomenon.

The combination of Forecasts
J.M. Bates

A seminal paper on the use of ensembling for time series forecasting. The idea that averaging models, rather than looking for a perfect model, was better for forecasting was both new and controversial to many traditional statisticians.

25 Years of Time Series Forecasting
J. De Gooijer and R. Hyndman

A thorough statistical summary of time series forecasting research in the 20th century from Professors Jan G. De Gooijer and Rob J. Hyndman.

A brief history of time series forecasting competitions
R. Hyndman

A shorter and more specific history from Professor Rob J. Hyndman which gives specific numbers, location, and authors of prominent time series forecasting competitions in the last 50 years.

Domain Specific Time Series Histories and Commentary

Weather Forecasting through the ages

A NASA history of how weather forecasting came to be, with emphasis on specific research challenges and successes in the 20th century.

Early Meteorological Data from London and Paris: Extending the North Atlantic Oscillation Series
R. Cornes

A doctoral thesis offering a fascinating account of the kinds of weather information available for two of Europe's most important cities, complete with extensive listings of the locations and nature of historic weather in time series format.

A Brief History of Medicine and Statistics
D. Mayer

A book chapter highlighting how the relationship between medicine and statistics depended greatly on social and political factors that made data and statistical training available for medical practitioners.

Random Time Series in Astronomy
S. Vaughan

A researcher's summary of the many ways time series analysis is relevant to astronomy and warnings about the danger of astronomers rediscovering time series principles or missing out on extremely promising collaborations with statisticians.

CHAPTER 2
Storing Temporal Data

Storage of time series data is necessary for almost every time series analysis problem. Overwhelmingly, the value of time series data comes in its retrospective, rather than live streaming, uses. For such cases, a good storage solution is one that enables greatest ease of access, reliability of data, and minimal computing resources invested in data storage. In this chapter we will discuss what aspects of a dataset should be considered when designing storage for that data. We will also discuss the advantages of SQL databases, no SQL databases, and a variety of flat file formats.

Designing a general time series storage solution is a difficult problem because there are so many different kinds of time series data, each with different storage, read/write, and analysis patterns. Some data will be stored and examined repeatedly, whereas other data is useful only for a short period of time after which it can be deleted altogether.

Here are a few use cases for time series storage that would have different read, write, and query patterns.

1. You are collecting performance metrics on a production system. You need to store these performance metrics for years at a time, although the older the data gets the less granular the data needs to be. Hence you need a form of storage that will automatically downsample and cull data as information ages.
2. You have access to a remote open source time series data repository, but you need to keep a local copy on your computer to cut down on network traffic. The remote repository stores each time series in a folder of files available for download on a web server, but you'd like to collapse all these files into a single database for simplicity. The data should be stored indefinitely and should be immutable as it is intended to be a reliable copy of the remote repository.

3. You created your own time series data by integrating a variety of data sources, at different time scales and with different preprocessing and formatting. The data collection and processing was laborious and time consuming. You'd like to store the data in its final form rather than running a preprocessing step on it repeatedly, but you'd also like the raw data in case you later decide to investigate preprocessing alternatives. You expect to revisit the processed and raw data often as you develop new machine learning models, refitting new models on the same data and also adding to your data over time as new, more recent raw data becomes available. You never want to downsample or cull data.

Notice that these use cases are quite varied in what their major demands on a system will be.

Importance of how performance scales with size

In the first use case, we would look for a solution which could incorporate automated scripting to delete old data. We would not be concerned with how the system scaled for large datasets because we would plan to keep the dataset small. In contrast, in the second and third use cases we would expect to have a stable, large collection of data (use case 2) or a large and growing collection of data (use case 3).

Importance of random vs. sequential access of data points

In use case 2 we would expect all of the data to be equally likely to be accessed as these time series data would all be the same "age" on insertion and would all reference interesting data sets. In contrast, in use cases 1 and 3 (and especially 1, given the description above) we would expect the most recent data to be accessed more often.

Importance of automation scripts

Use case 1 seems as though it might be automated, whereas use case 2 would not have any automation since it would be immutable. Use case 3 suggests little automation but a great deal of data fetching and processing of all portions of the data, not just the most recent. In use case 1 we would want a storage solution that could be integrated with scripting or stored procedures, whereas in use case 3 we would want a solution that permitted a great deal of easy customization of data processing.

Just these three examples already offer a good idea of the many use cases a time series solution, if general, needs to meet. In real use cases you will be able to tailor your storage solution to a specific use case and not need to worry about finding a tool that fits all use cases. That said, you will always be choosing from a similar range of available technologies, and these tend to boil down to

- SQL databases
- NoSQL databases
- flat file formats

Despite the general popularity of NoSQL solutions, as far as media, blogging, and conference presentations go, the organizational reality is that many companies and research institutes continue to use SQL and file solutions for time series data. This will likely change as NoSQL continues to expand its role in many organization's technology use cases, but for now you will often inherit SQL databases or files if you come into an existing storage solution.

In this chapter, we aim to cover all three options and to discuss the advantages and disadvantages of each. Of course the specifics will always depend on the use case at hand, but this chapter will equip you with the first line of questions when you begin your search for a time series storage option that works for your use case.

We will first discuss what questions you should ask at the outset when picking a storage solution. Then we will look at the great SQL vs. NoSQL debate and examine some of the most popular time series storage solutions. Finally we will consider setting up policies to let old time series data expire and eventually be culled or deleted altogether.

Defining Requirements

When you are thinking about storage for time series data, you need to ask yourself a variety of questions. This inventory below will help you get started.

- How much time series data will you be storing? How quickly will that data grow?
  - — You want to choose a storage solution appropriate to the growth rate of the data you expect. Database administrators moving into time series work from transaction-oriented data sets are often surprised at just how quickly time series data sets can grow.
- Are your measurements more like endless channels of updates (e.g. a constant stream of web traffic updates) or more like distinct events (e.g. an hourly air traffic time series for every major US holiday in the last 10 years)
  - — If your data is like an endless channel, you will mostly look at recent data. On the other hand, if your data is a collection of distinct time series of distinct events, events more distant in time may still be fairly interesting. In this latter case, random access is a more likely pattern.
- Will your data be regularly or irregularly spaced?
  - — If your data is regularly spaced, you can calculate more accurately in advance how much data you expect to collect and how often that data will be inputted. If your data is irregularly spaced, you must prepare for a less predictable style of data access that can efficiently allow for both dead periods and periods of write activity.
- Will you continuously collect data or is there a well-defined end to your project?

— If you have a well-defined end to the data collection, this makes it easier to know how large a data set you need to accommodate. However, many organizations find that once they start collecting a particular kind of time series, they don't want to stop!

- What kind of use will you be making of your time series? Do you need real time visualizations? Preprocessed data for a neural network to iterate over thousands of times? Sharded data highly available to a large mobile user base?
  — Your primary use case will indicate whether you are more likely to need sequential or random access to your data and how important a factor latency should be to your storage format selection.
- How will you cull or downsample data? How will you prevent infinite growth? What should be the lifecycle of an individual data point in a time series?
  — It is impossible to store all events for all time. It is better to make decisions about data deletion policies systematically and in advance rather than on an ad hoc basis. The more you can commit to up front, the better a choice you can make regarding storage formats. More on this in the next section.

The answers to these questions will indicate whether you should store raw or processed data, whether the data should be located in memory according to time or according to some other axis, and whether you need to store your data in a form that makes it easy to read data as well as write data. Use cases will vary, so you should do a fresh inventory with every new data set.

Live Data vs. Stored Data

When thinking about what storage options are desirable for your data, it's key to understand the life cycle of your data. The more realistic you can be about your actual use cases for data, the less data you will need to save and the less time you will need to worry about finding the optimal storage system because you won't quickly size up to an intractable amount of data. Often organizations over-record events of interest and are afraid to lose their data stores, but having more data stored in an intractable form is far less useful than having aggregated data stored at meaningful time scales.

For short-lived data, such as performance data that will only be looked at to ensure nothing is going wrong, it's possible that you may never need to store the data in the form in which it's collected, and not for very long. This is particularly true for event-driven data, where no single event is important.

Suppose you are running a web server which records and reports to you the amount of time it took every single mobile device accessing a web page to fully load that web page. The resulting irregularly spaced time series might look something like this:

Timestamp	Time to load page
April 5, 2018 10:22:24 pm	23s
April 5, 2018 10:22:28 pm	15s
April 5, 2018 10:22:41 pm	14s
April 5, 2018 10:23:02 pm	11s

For many reasons, you would likely never be interested in any individual measurement of the time to load a page. You would always find this information most useful in the aggregate, and even then, only for a short time. Suppose you were on call overnight for that server. You'd want to make sure you could show that performance was good while you were responsible. You could simplify this to a datapoint for your 12 hours of being on call, and you would have most of the information you could ever want.

Time slice	Most popular hour of access	Num loads	Mean time to load	Max time to load
April 5, 2018 8pm - 8 am	11 pm	3,470	21s	45s

In such a case, you should never plan to indefinitely store the individual events. Rather, you should build a storage solution that provides for staging of the individual events only as temporary storage until the data goes into its ultimate form. You will save yourself and your colleagues a lot of grief by preventing runaway data growth before it can even get started. Instead of having 3470 individual events, none of which interests anyone, you will have readily accessible and compact figures of interest that are easily accessible and understood.

Even for longer lived data or data that is less granular such that individual data points have more importance than a single mobile device website access, you should make every effort to simplify data storage via aggregation and deduplication whenever possible. Below we consider a few opportunities for reducing data without losing information. You will doubtless find many more when you look at your own time series data sets.

Slowly-changing variables

If you are storing a state variable, consider only recording those data points where the value has changed. For example, if you are recording temperature at five minute increments, your temperature curve, particularly if you only care about a value such as the nearest degree, might look like a step function. In such a case, it's not necessary to store repetitive values, and you will save storage space by not doing so.

Noisy, high frequency data

If your data is noisy, there are reasons for not caring very much about any particular data point. You might consider aggregating data points before recording them since the high level of noise makes any individual measurement less valuable. This will of course be quite domain specific and you will need to ensure that downstream users are still able to assess the noise in the measurements for their purposes.

Stale data

The older data is, the less likely your organization is going to use that data, except in a very general way. Whenever you are beginning to record a new time series data set, you should think preemptively about when the time series data is likely to become irrelevant.

- Is there a natural expiration date?
- If not, can you look through past research from your analytics department and see how far back they go realistically? When did any script in a git repository actually access the oldest data in your data set?

If you can automate data deletion in a way that will not impede data analysis efforts, you will improve your data storage options by reducing the importance of scalability or the drag of slow queries on bloated data sets.

Legal considerations (Really!)

If you are working in a large organization or an organization that works on data likely to be required by outside parties, such as for regulatory audits, this should actor into your system requirements. You should assess the likelihood of how often you are likely to get requests from outside parties for data, how much data would be required in such cases, and whether you would be able to choose the format of the data or whether regulations impose a specific form or overall structure to what you'd need to provide.

It may seem unduly cautious to cite legal requirements when assessing data storage specifications, but recent developments in Europe and the United States suggest increasing interest by law makers regarding the datasets used to power machine learning in modern consumer-facing applications. Most importantly, the European Union's General Data Protection Regulation (GDPR) imposes strong requirements on organizations that build machine learning models using data they store about individuals. You will want to make sure that your organization is able to comply with this law, particularly because the concrete specifics of how the law will be applied are still somewhat unclear, so that you need to plan for change as well as planning to comply with existing rules.

So far, we have discussed the general range of use cases for time series storage. We have also reviewed a general set of queries related to how a time series data set will be produced and analyzed so that these queries can inform our selection of a storage format. We will now review the two major options for storing time series: databases and files.

Database Solutions

For almost any data analyst or data engineer, a database is an intuitive and familiar solution to the question of how to store data. As with relational data, a database will often be a good storage choice for time series data. This is particularly true if you want an out-of-the-box solution with any of the classic database characteristics listed below:

- a storage system that can scale up to multiple servers
- a low latency read/write system
- functions already in place for computing commonly used metrics (such as computing the mean in a group-by query, where the group-by can apply to time metrics)
- troubleshooting and monitoring tools that can be used to tune system performance and analyze bottlenecks

These are among many good reasons to opt for a database rather than a file system, and you should always consider a database solution for data storage, particularly when working with a new dataset. A database, particularly a NoSQL database, can help you preserve flexibility. Also a database will get your project up and running sooner than if you are going to work with individual files because much of the boilerplate you will need is already in place. Even if you ultimately decide on a file storage solution (and most people won't), working with a database first can help you determine how to structure your own files when your new data process matures.

In the remainder of this section, we will cover the respective advantages of SQL and NoSQL databases for time series and then discuss currently popular database options for time series applications.

The good news is that time series graphs appear to be the fastest growth category of database at present so that you can expect to see more and even better options for time series database solutions over time.

SQL vs NoSQL

SQL vs NoSQL is just as lively a debate in the time series database community as it is in the wider database sector. Many expert database administrators insist that SQL is the only way to go and that there is no data in any shape that cannot be well described

by a good set of relational tables. Nonetheless, in practice there is often a drop in performance when organizations try to scale SQL solutions to accommodate large amounts of time series data, and for this reason it's always worth considering a NoSQL solution as well, particularly if you are seeking an open-ended solution that can scale to accommodate cases where time series data collection is commenced with no finite time horizon in sight for the data collection.

While both SQL and NoSQL solutions can be good for time series data, we first motivation our discussion of the difficulties of applying database logic to time series data by discussing how time series data differs from the kind of data for which SQL databases were originally developed.

Characteristics of the data that originally inspired SQL databases

We can best understand the mismatch between SQL like thinking and time series data by understanding the history of SQL solutions. SQL solutions were traditionally based on transactional data. Transactional data is whatever data is needed to completely describe a discrete event. A transaction is composed of attributes that reflect many primary keys, such as product, participants, time, and value of a transaction. Notice that time can be present as a primary key but only as one among many, not as a privileged axis of information.

There are two important features of transactional data that are quite different from time series needs.

1. In transactional data, existing data points are often updated.
2. In transactional data, the data are accessed somewhat randomly as there is no necessary underlying ordering.

Characteristics of time series data

Time series data elaborates the entire history of something, whereas a transaction recording only tells us the final state. Hence time series data does not generally require updating and does not generally have data/memory randomly accessed because it provides a series of slices to document the evolution of a situation rather than a single most-up-to-date point.

This means that the performance objectives that have been key to decades of SQL database design are not very important for time series databases. In fact, when we consider objectives in designing a database specifically for time series, we have quite different priorities because of how we will use our time series data. The primary features of time series data use case are:

1. Write operations predominate over read operations (usually by a large factor)
2. Data is neither written nor read nor updated in random order but rather in the order related to temporal sequencing.

3. Concurrent reads are far more likely than they are for transaction data
4. Few, if any, primary keys other than time itself.
5. Bulk deletes are far more common than individual data point deletion.

It is these features that support the use of a NoSQL database because many general application NoSQL databases offer much of what is desirable for time series databases, particularly the emphasis on write operations over read operations. NoSQL databases also tend to map well conceptually to time series data, in that they natively reflect aspects of time series data collection, such as that not all fields are collected for all data points. The flexible schema of NoSQL can feel very natural with time series data. In fact, I would argue that much of the data that motivated the current popularity of NoSQL databases is time series data.

Also, for this reason, out of the box NoSQL databases tend to be more performant than SQL databases for write operations. Below I plot performance of a commonly used SQL database vs. a commonly used NoSQL time series database for the case of data point insertions (write operations). It is this out-of-the-box performance that can be very helpful for many time series applications.

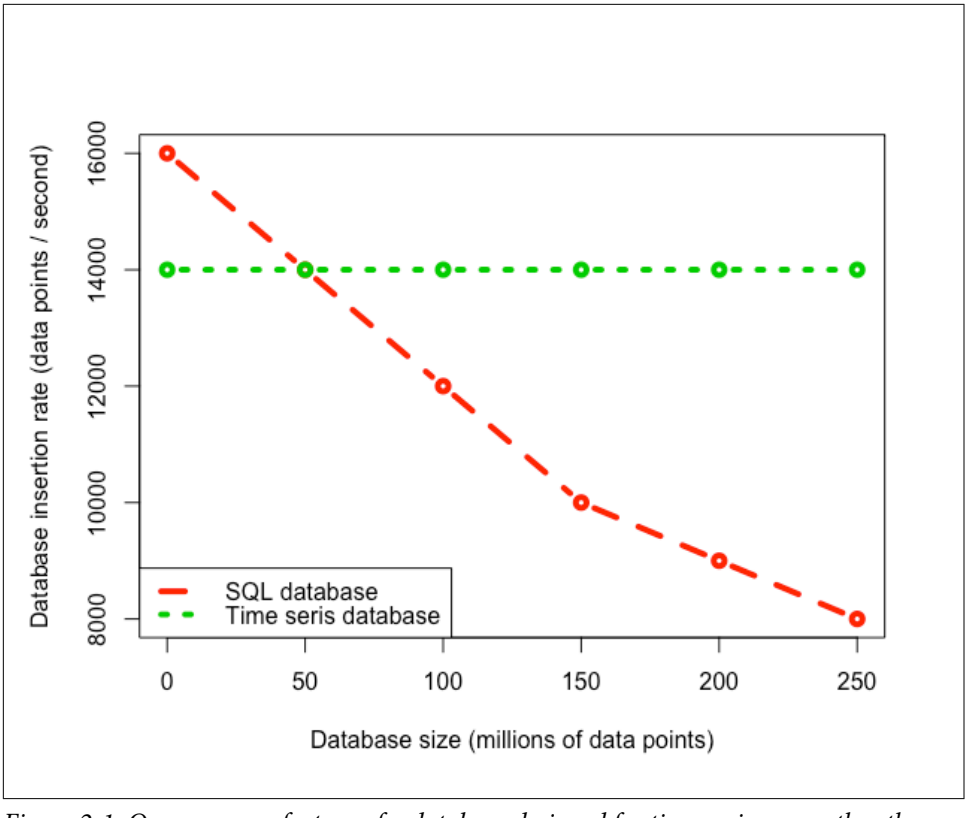

Figure 2-1. One common feature of a database designed for time series use rather than a traditional all-purpose SQL database is that the data insertion rate is constant with respect to the database size rather than decreasing with database size.

How to choose between SQL and NoSQL

It may seem that I am attempting to push all readers towards the NoSQL use case, but there are also many good use cases of SQL databases. When thinking about your own data, you should keep in mind the principle that always applies to data storage, be it in a SQL database, a NoSQL database, or a plain old text file: **data that tends to be requested at the same time should be stored in the same location**. That's the most important factor regardless of your use case.

Many conference presentations and blog posts herald NoSQL solutions as precisely catering to the situation of a high-write low update scenario where data is not ordered randomly. Nonetheless, SQL databases are a viable and regularly used time series storage option. In particular, with some architectural changes in emphasis to traditional SQL databases and their internal memory structure, these challenges can be met while keeping some of the advantages of SQL. For example, something as simple

and seemingly obvious as structuring SQL tables internal memory representation to account for time can make a substantial performance difference.

Ultimately, the distinctions between NoSQL and SQL are highly implementation dependent and not as systematic or important as they are made out to be. Let your data drive your selection of a specific implementation of one of these technologies. As you consider the attributes of your time series data and access use patterns, you can keep some general limitations in mind.

Pluses of SQL for time series

- If your time series is stored in a SQL database, you can easily relate it to relevant cross-sectional data also stored in that database.
- Hierarchical time series data is a natural fit with relational tables. An appropriate set of SQL schema will help you group related time series and clearly delineate the hierarchy, whereas these could be easily scattered less systematically in a NoSQL solution.
- If you are crafting a time series based on transactional data, where the latter is best stored in a SQL database, it will be advantageous to also store your time series in the same database for ease of verification, cross referencing, etc.

Pluses of NoSQL for time series

- Fast writes
- Good if you don't know enough about future data to design intelligent and robust schema
- For the inexpert user, these databases are often the more performant out-of-the-box solution because you are less likely to design awkward schema or to be locked into a mediocre schema design

Popular Time Series Database and File Solutions

Below is a short discussion of some popular database solutions for time series data. These give you a flavor of what is available in addition to traditional SQL solutions. Note that technologies discussed here occupy a crowded and fragmented technology landscape. What is commonly used this year may not be so popular next year. For this reason, this discussion should not be viewed as a specific technology recommendation so much as a set of samples helping to illustrate the current state of the market.

Time series specific databases and related monitoring tools

First we discuss tools that are built specifically for the purpose of storing and monitoring time series data. In particular we take a look at one technology that is specifi-

cally a time series database (InfluxDB) and another product that is a performance monitoring tool which can double as a time series storage solution (Prometheus). The advantages of each tool necessarily reflect their distinct emphases and use patterns.

InfluxDB. InfluxDB is unabashedly a time series database, per its own description on the Github project's webpage:

*InfluxDB is an open source **time series database** ... useful for recording metrics, events, and performing analytics.*

In InfluxDB, data is organized by time series. A data point in InfluxDB consist of

- a timestamp
- a measurement label indicating what the measurement consists of
- one or more key-value fields (such as `temperature=25.3`)
- key-value pairs containing metadata tags.

InfluxDB, as a time aware database, automatically timestamps any data point that comes in without a timestamp. Also InfluxDB uses SQL like querying, such as queries like the following

```
SELECT * FROM access_counts WHERE value > 10000
```

Other advantages of InfluxDB are:

- Data retention options that allow you to easily automate the designation and deletion of stale data of data
- High data ingestion speed and aggressive data compression
- Individual time series can be tagged to allow for fast indexing of time series that match a specific criterion
- Membership in the mature TICK stack, which is a platform for capture, storage, monitoring, and displaying of time series data.

There are many other time series specific databases - right now InfluxDB happens to be the most popular, so it is the one you are most likely to encounter. The options it offers are those commonly offered by time series specialized databases, as these options reflect the most commonly desired attributes for storage of time series data.

As a database, InfluxDB is a pushed based system, meaning that when you use this system you push data into the database for ingestion. This is different from the next option which will be discussed.

Given these specifications, a solution such as InfluxDB provides all general functionality right out of the box in a time-aware manner. This means you can use your existing SQL skills while also benefiting from advantages related to the need to capture

but control the growth of time series data. Finally, you have the fast write so necessary when capturing time series data as it is produced.

Prometheus. Prometheus describes itself as "monitoring system and time series database" that works via HTTP. This description indicates the general emphasis, which is first on monitoring and second on the related task of storage. The great advantage of Prometheus is that it is a pull-based system, which means that the logic for how often and how data is polled for creating a time series is centralized and be easily adjusted and inspected.

Because Prometheus works as a pull-based system, it is a great resource to have during emergencies as it is atomic and self-reliant. However, Prometheus is also not guaranteed to be entirely up-to-date or accurate, due to this pull-based architecture. While it should be the go-to technology for quick-and-dirty performance monitoring, it is not appropriate for applications where data has to be 100% accurate.

Prometheus uses a functional expression language called PromQL for its querying.

```
access_counts > 10000
```

Prometheus also offers, via this language, an API for many common time series tasks, even including sophisticated functions such as making a prediction (`predict_linear()`) and calculating the per-unit-of-time rate of increase of a time series (`rate()`). Aggregation over periods of time is also offered with a simple interface. Prometheus tends to emphasize monitoring and analysis over maintenance, so compared to InfluxDB there are fewer automated data-curating functionalities.

Prometheus is a helpful time series storage solution particularly for livestreaming data and data where availability is a paramount consideration. It has a steeper learning curve due to having its own script and the less database-like architecture and API but is widely used and enjoyed by many developers.

General NoSQL databases

While time series specific databases offer many advantages, you can also consider the case of a more general use NoSQL databases. These sorts of databases are based on a 'document' structure rather than a table structure, and they do not usually have many explicit functions specifically for time series.

Nonetheless, the flexible schema of NoSQL databases are particularly useful for time series data, particularly for new projects where the rhythm of data collection and the number of input channels may change through the course of a dataset's lifetime. For example, initially a time series may start out as only one channel of data but gradually grow to include more kinds of data all timestamped together. Later, it may be decided that some of the input channels are not particularly useful, and they may be discontinued.

In such a case, storing the data in a SQL table would be difficult for several reasons, and would result in a great many NaNs where data was not available. A NoSQL database would simply have the channels absent when they were not available rather than needing to mark up a rectangular data store with many NaNs.

One popular and performant NoSQL time series database is MongoDB. Mongo is particularly aware of its value as a time series database and has a strong push to develop IoT friendly architecture and instructions. It offers high level aggregation features that can be applied to aggregation operations by time and time-related groupings, and it also offers many automated processes to divide time into human-relevant markings, such as day of the week or day of the month:

- $dayOfWeek
- $dayOfMonth
- $hour

Also, Mongo has devoted extensive documentation efforts to demonstrating how time series can be handled with Mongo (such as here, here, and here but see here). A ready set of documentation and institutional focus on time series data explicitly mean that users can expect this database to continue developing even more time-friendly features.

More useful than all this functionality, however, is Mongo's flexibility in evolving schema over time. This schema flexibility will save you a lot of grief if you work on a rapidly evolving time series dataset with ever-changing data collection practices.

For example imagine a sequence of time series data collection practices such as what might occur in a healthcare startup.

- Your startup launches as a blood pressure app and collects only two metrics, systolic and diastolic blood pressure, which you encourage users to measure several times daily...
- But your users want enhanced lifestyle advice and they are happy to give you more data. Users provide everything from DOB to monthly weight recordings to hourly step counts and calorie counts. Obviously, these different kinds of data are collected at drastically different rhythms....
- But later you realize some of this data is not very useful so you stop collecting it...
- But your users miss your display of the data even if you don't use it, so you restart collection of the most popular data ...
- And then the government of one of your largest markets passes a law regarding the storage of health data and you need to purge that data or encrypt, so you need a new encrypted field...

- And changes continue

When you need the querying and schema flexibility described in an application like the one above, you will be well served with a general NoSQL database that offers a reasonable balance between time-specific functionality and more general flexibility.

Another advantage of a general-purpose NoSQL database rather than a time series specific database is that you can more easily integrate non-time series data into the same database for ease of cross-referencing the related data sets. Sometimes a general purpose NoSQL database is just the right mix of performance considerations and SQL like functionality without the cleverness needed to optimize a SQL database schema for time series functionality.

In this section we have examined NoSQL database solutions[1] and surveyed some options now in popular use. While we can expect the particular technologies that dominate the market to evolve over time, the general principles on which these technologies operate, and the advantages they offer, will remain the same. As a quick recap, some of the advantages we find in different varieties of databases are:

- High read or write (or both) capacities
- Flexible data structures
- Push or pull data ingestion
- Automated data culling processes

A time series specific database will offer the most out-of-the-box automation for your time series specific tasks, but it will offer less schema flexibility and fewer opportunities to integrate related cross-sectional data than will a more general all-purpose NoSQL database.

More generally, databases will offer more flexibility than flat file storage formats, but this flexibility means that databases are less streamlined and less I/O performant than simple flat files, which we discuss next.

File Solutions

At the end of the day a database itself is a piece of software that integrates both scripting and data storage. It is essentially a flat file wrapped in a special piece of software responsible for making that file as easy to use and safe as possible.

Sometimes it makes sense to take away this outer layer and take on the full responsibility of the data storage ourselves. While this is not very common in business applications, it is commonly done in scientific research and also done in the rare industrial

1 The optimization of SQL databases for time series data is beyond the scope of this book and tends to be very specific to use case. Some resources at the end of the chapter address this question.

application (such as high frequency trading) in which speed is paramount. In such cases, it is the analyst who will design a much more intricate data pipeline that involves allocating storage space, opening files, reading files, closing files, protecting files, etc, rather than simply writing some database queries.

A flat file solution is a good option if any of these conditions apply

- your data format is mature so that you can commit to a specification for a reasonably long period of time
- your data processing is I/O bound so that it makes sense to spend development time speeding this up
- you don't need random access but can read data sequentially

In this chapter we briefly survey some comm flat file solutions. You should also remember that you can always create your own file storage format, although this is fairly complex and is usually only worth doing if you are working in a highly performant language, such as C++ or Java.

If you are in the situation that your system mature enough and performance sensitive enough to suit a flat file system, there are several advantages to implementing a flat file system, even if it may mean the chore of migrating your data out of a database. These advantages include:

- A flat file format is system agnostic. If you need to share data, you simply make files available in a known shared format. There is no need to ask a collaborator to remotely access your database or set up or her own mirrored database.
- A flat file format's I/O overhead is necessarily less than that for a database because this amounts to a simple read operation for a flat file rather than a retrieval and a read, as it would be for a database.
- A flat file format encodes the order in which data should be read, whereas not all databases will do so. This enforces serial reading of the data, which can be desirable, such as for deep learning training regimes.
- Your data will occupy a much smaller amount of memory than it would on a database because you can maximize compression opportunities. Relatedly you can tune the degree of data compression explicitly to balance between your desire to minimize data storage footprint and minimize I/O time in your application. More compression will mean a smaller data footprint but longer I/O waits.

Numpy and Pandas File Formats

Numpy

If your data is purely numeric, one widely used option for holding that data is a Python numpy array. Numpy arrays can easily be saved in a variety of formats, and there are many benchmarking publications comparing the relative performance of these formats. For example, take a look at this Github repo designed to test efficiency and speed of various numpy file formats.

A downside of numpy arrays is that they have a single data type, which means you cannot store heterogeneous time series data automatically but must think about whether having just one data type can work for your data in raw or processed form (although there are ways around this restriction). Another downside of numpy arrays is that it is not natural to add labels to rows or columns, so that there is no straightforward way to, for example, timestamp each row of an array.

The advantage of using a numpy array is that there are many options for saving this data structure, including a compressed binary format that takes up less space and has faster I/O than you will see with a database solution. It is also as out-of-the-box as it gets in terms of a performant data structure for analysis as well as storage.

Pandas

If you want easy labeling of data or easy storage of heterogenous time series data (or both options), consider the less streamlined but more flexible pandas Data Frame. This can be particularly useful where we have a time series that consists of many distinct kinds of data, perhaps including event counts (int), state measurements (floats) and labels (strings or one hot encodings). In such a case you will likely want to stick to Pandas data frames. (Also remember that the name pandas actually comes from an elision of "panel data", so this is a natural format for many use cases.)

Pandas data frames are widely used, and so there are several resources online to compare a variety of formats used to store these data.

Standard R equivalents

The native formats for storing R objects are `.Rds` and `.Rdata` objects. These are binary file formats, so they will necessarily be more efficient both for compression and I/O than text-based formats, and in this way are akin to Python's pandas Data Frame storage. Relatedly the `feather` format is a format that can be used in both R and Python to save Data Frames in a language agnostic file format. For R users, the native binary formats will (of course) be the most performant.

Format Name	Relative Size	Relative Time to load
.RDS	1x	1x
feather	2x	1x
csv	3x	20x

As we can see from the graph above, the native format is a clear winner to minimize both storage space and I/O, and those who use a text-based file format rather than binary will pay a heavy price both for storage and I/O slowdowns.

Xarray

When you get to having time series data that is many dimensions, it may be time to think about a more industrial data solution, namely `xarray`. Xarray is very useful for a number of reasons.

- Named dimensions
- Vectorized mathematical operations, like numpy
- Group by operations, like pandas
- Database-like functionality that permits indexing based on a time range
- A variety of file storage options

Xarray is a data structure supports many time series specific operations, such as indexing on time, resampling on time, interpolating data, and accessing individual components of a date time. Xarray was built as a high-performance scientific computing instrument, and it is woefully underutilized and under-appreciated for time series analysis.

Xarray is a data structure in Python, and it comes with a choice of storage options. Xarray implements both `pickling` and another binary file format called `netCDF`, a universal scientific data format supported across many platforms and languages. If you are looking to "up" your time series game in Python, a good place to start is Xarray.

There are many options for flat file storage of time series data, some with associated functionality (xarray) and some as very streamlined purely numeric formats (numpy). When migrating a dataset from a database-oriented pipeline to a file-oriented pipeline there will be some growing pains related to simplifying data and rewriting scripts to move logic out of the database and into the explicit ETL scripts. In cases where performance is paramount, moving your data into files is likely the most important step to reduce latency. Common cases where this is desirable include:

- latency sensitive forecasting, such as for user-facing software
- I/O intensive repetitive data access situations, such as training deep learning models

On the other hand, for many applications the convenience, scalability, and flexibility of a data base is well worth the higher latency. Your optimal storage situation will

depend very much on the nature of the data you are storing and what you want to do with it.

More Resources

Comments on the State of the Time Series Database Technologies

Thoughts on Time-series Databases
J. Moiron

This classic blog post from 2015 provides a glimpse into an earlier time period with time series databases and the cult of recording everything. This high level overview of options for storing time series databases and typical use cases is highly informative for those just beginning to understand database administration and engineering.

List of Time Series Databases
P. Jinka

This page is frequently updated and maintains a lengthy list of time series databases currently on the market. Each database entry comes with highlights as to how a particular database relates to its competitors and predecessors.

Percona Blog Poll: What Database Engine Are You Using to Store Time Series Data?
P. Zaitsev

A 2017 poll of database engineers to determine what is actually happening on the industry level. This poll indicated that relational databases (SQL databases) continued to dominate as a group, with 35% of respondents indicating using these databases to store time series data. ElasticSearch, InfluxDB, MongoDB, and Prometheus were also favorites.

The State of the Time Series Database Market
R. Stephens

This recent (2018) data-driven write up by a tech analyst describes the most popular solutions to time series database storage via an empirical investigation of Github and Stackoverflow activity. The report also indicates a high degree of fragmentation in the time series storage domain due to a wide variety of use cases for time series data and also due to a larger trend towards segmentation in databases generally.

Prometheus documentation: Comparison to Alternatives

This extremely detailed and comprehensive list compares Prometheus to a number of other popular time series storage solutions. You can use this model as a quick reference for the dominant data structures and storage structures used by the alternatives

to Prometheus. This is a good place to get an overview of what's on offer and what the trade-offs are between various databases designed for temporal data.

Adapting General Database Technologies to Time Series Use Cases

Storing Time Series in PostreSQL Efficiently
G. Trubetskoy

An older (2015) but still relevant blog post as to how to store time series in Postgres in a performance way. The blog explains the difficulties of the "naive" approach of a column of values and a column of timestamps and gives practical advice for approaches based on Postgres arrays.

Using Redis as a Time Series Database
J. Carlson

This article provides detailed advice and examples related to using Redis as a time series database. While Redis has been used for time series data since its creation, there are several gotchas the author points out as well as advantageous data structures that can be applied in many time series use case scenarios. This article is useful both specifically for learning more about how to use Redis with time series data but also as an example of how more general tools can be repurposed for time-series specific uses.

Time Series Data: Why (and how) to use a relational database instead of NoSQL
M. Freedman

This blog post by TimescaleDB's founder describes the ways in which his team built a time series database as a relational database with memory layout modifications specific to temporal data. TimescaleDB's contention is that the main problem with traditional SQL databases for time series data is that such databases do not scale, resulting in slow performance as data is shifted into and out of memory to perform time-related queries. TimescaleDB propose laying out memory and memory mappings to reflect the temporal nature of data and reduce swapping disparate data into and out of memory.

About the Author

Aileen is a software engineer and data analyst with a data background that runs the gamut from experimental physics to healthcare startups to finance. She is a frequent speaker at industry conferences, and she has given talks and tutorials around the world on the many challenges of time series analysis.