Contents

Chapter 1

Introduction

General advice: - no contractions - cite statement sources - More helping figures

1.1 Data Analysis

Data analysis is the act of analyzing, cleansing, manipulating, and modeling data in order to identify usable information, generate conclusions, and help decision-making. [1] Data analysis has several dimensions and approaches, including a wide range of techniques known by various names and applied in a variety of business, science, and social science sectors. [2] In today's corporate world, data analysis plays an important part in making decisions more scientific and assisting firms in operating more efficiently. [3] Data mining is a type of data analysis technique that focuses on statistical modeling and knowledge discovery for predictive rather than purely descriptive purposes, whereas business intelligence is a type of data analysis that focuses on aggregation and is primarily concerned with business information, more on this later.

1.1.1 Procedure for analyzing data

The term *"analysis"* refers to the process of breaking down a whole into its constituent parts for closer evaluation. Data analysis is the act of getting raw data and then transforming it into information that users can utilize to make decisions. Data is gathered and processed in order to answer questions, test hypotheses, or refute theories.[11]

There are various distinct phases that can be identified, these are iterative in the sense that input from later phases may lead to further effort in earlier ones. Similar stages can be found in the CRISP framework, which is used in data mining.

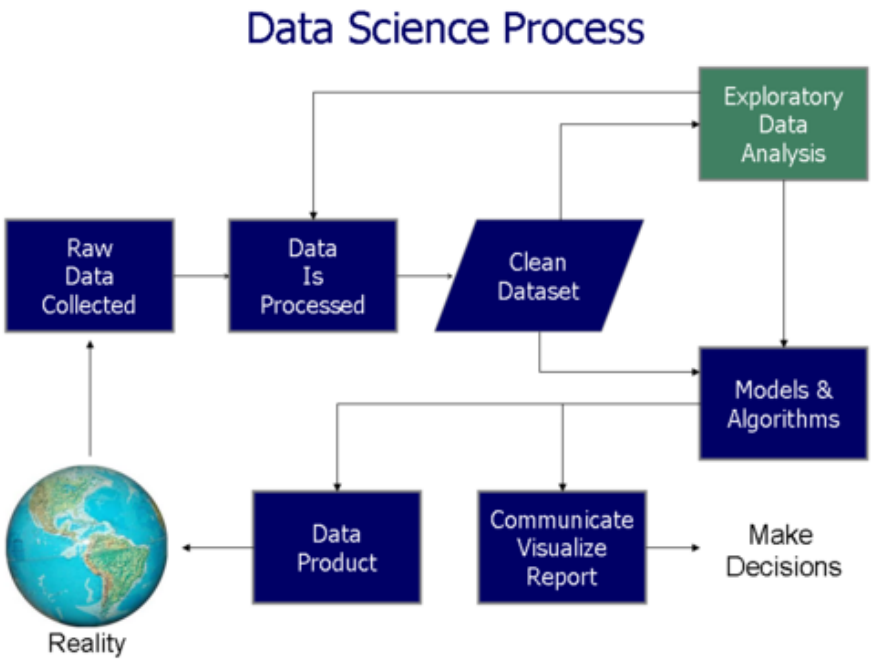


Figure 1.1: Data science process flowchart [**Book:doing_data_science**]

1.1.2 What is Data?

We talked a lot about data in the last section and while it is important that we can analyze and understand data, but what is data? Understanding what data is it's a prerequisite for being able to use it properly, perhaps the most important thing as far as we're concerned

1.1.3 Data Cleaning

1.1.4 Data Transformation

1.1.5 Data Reduction

1.2 Operational environment: Maintenance

not all data are the same, treated differently depending on the context. Increase the awareness of the problem

1.2.1 Predictive maintenance

[**Misc:zensor_blog**]

1.2.2 Prescriptive maintenance

1.2.3 Repair minimization

and mention the company at the end.

1.2.4 A possible solution

Today, most often, technical data sheets coupled with the knowledge of a number of unique experienced individuals are used to determine when the asset requires maintenance. Product quality only becomes an issue when customers start complaining and repairs are done when it's already far too late. All of these puts tremendous strain on the people responsible for the asset, while it could be avoided. Unexpected shutdowns are costly and very demanding for the workforce involved; a possible solution would be making assets smart in order to increase the availability. The only way to validate the actual health is by having a continuous look at a broad data set and adding a specific multi-aspect monitoring setup consisting of different sensor types that follow the behaviour of the asset's general state-of-health.

Chapter 2

The hosting company: Zensor

2.1 Why Zensor exist?

Today, most often, technical data sheets coupled with the knowledge of a number of unique experienced individuals are used to determine when maintenance is required for the asset. Product quality only becomes an issue when customers start complaining and repairs are done when it's already far too late. All of these puts tremendous strain on the people responsible for the asset, while it could be avoided. Unexpected shutdowns are costly and very demanding for the workforce involved; a possible solution would be making assets smart in order to increase the availability. The only way to validate the actual health is by having a continuous look at a broad data set and adding a specific multi-aspect monitoring setup consisting of different sensor types that follow the behaviour of the asset's general state-of-health.

2.1.1 About the company

Zensor [**Misc:zensor_official_website**] provides full, integrated and intelligent monitoring solutions for the industrial production, renewable energy and infrastructure sectors. This allows to Zensor's customers to convert the potential contained in the world of IoT and Industry 4.0 into their reality. The company enable digitalization, but with an interface oriented towards the real human: an expert solution without the need for internal experts. It provides not only monitoring devices or data analysis, but offers a full, standardized and standalone end-to-end product that leverages the value contained in a subset of the following aspects:

- operational efficiency
- predictive maintenance
- energy efficiency
- ageing and degradation

- safety

A standard offering consists in several aspects (if required) ordered logically below; as such Zensor takes end-to-end responsibility in monitoring the health and efficiency of structures and processes.

1. Hardware {sensors and acquisition units}
2. Installation and Commissioning {engineering and CAD}
3. Data Management {data transfer, storage, coupling to existing data sources (SCADA, weather, operational...), data cleaning and treatment}
4. Analysis and Reporting {predictions, trend and event detection, real-time reporting through online dashboards.}

Table 2.1: Assets, Industries, and Infrastructure for which Zensor has specific products

Asset	Industries	Infrastructure
Rolling cranes	Metal Production	Offshore wind
Grinders and crushers	Mining and Materials	Rail
Flattener rollers	Food Production	Civil Infrastructure
Rolling mills	Glass Production	Energy
Conveyor belts	Discrete Manufacturing	
Tunnels	Textiles	
Chain transporters		
Sieves		
Bridges		

2.1.2 Philosophy

Vision Technological advance can only bring real value to society when the user-facing component is driven by Simplicity and Clarity for the end-user. Zensor sees this as the fastest and most certain route to a world where man-made structures affect the sustainability of our planet in the least possible way:

1. they are intrinsically safe;
2. their useful life is optimized to the maximum;
3. their impact on environment and society is quantified and communicated.

Mission Translate technological innovations in monitoring and analysis into easy-to-understand, tangible and relevant information that we share in the way tailored for either production managers, management, or maintenance professionals ... As such, we are the knowledgeable and easy-to-reach companion for owners and operators in making their assets increasingly safe, efficient and sustainable. If up to us, till eternity.

2.2 Zensor Approach

2.2.1 What are the stages

$$Hardware \rightarrowtail Installation \rightarrowtail DataManagement \rightarrowtail Analysis$$

Hardware and/or other Existing Data sources Sometimes not enough data is available from the beginning. Deriving valuable insights from a monitoring system states from the data: identifying and locating the relevant data in existing databases/data warehouse or putting the right sensors, with appropriate settings, on the right positions and measurement conditions; afterwards reading them out in the optimal way. All of this are defined clearly in every asset-specific package.

Installation and Commissioning Push the button of Industry 4.0. Initially links to the existing data sources are established and data gets ingested. Where required a set of acquisition units and sensors is installed on the machine or structure. After a final verification on the spot (SAT) the monitoring system is launched: the assets enter the IoT.

Data Management Data is continuously streaming in from individual setups as well as historian sources. Structuring, verifying and cleaning the data sets is an essential prerequisite to allow for a profound analysis afterwards: on your way to an automated, continuous and smart follow-up.

Analysis and Clear Reporting Advanced insights are unlocked using algorithms based on physics as well as big data approaches. Clear dashboards, warnings and periodic reports inform the owner or line manager about the present state and upcoming issues. Surprises are avoided, standstills reduced.

2.2.2 Advanced features and metrics

Here is a non-exhaustive list of the main monitoring metrics available

Availability Have a continuous idea of availability, automatically as the platform combines different input streams and contextual information.

Performance Based on the data collected and machine-learning based methods for determining the operational condition the performance is calculated.

Warnings Whenever values start to deviate, or data streams stop, warnings are sent. This avoids 'black holes' in the insights of the production line or assets.

Quality Coupling to existing databases or using human input fields the product quality is linked to operational process parameters.

MTTF The $MeanTimeTillFailure$ is tracked continuously, for each asset covered the overall 'disturbance free' operation is displayed.

MTBF As events and operating conditions are automatically detected the $MeanTimeBetweenFailures$ is determined continuously, giving a good insight on where optimization is possible.

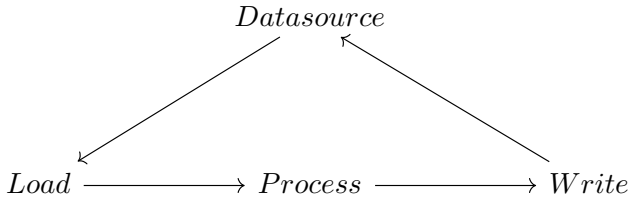
OEE Using all parameters cited above the $OverallEquipmentEffectiveness$ is determined, a major parameter for optimizing asset management strategies and future investments, with a huge cost savings potential.

Factory information systems Such systems are crucial for obtaining operational excellence. When well managed they maximize efficiency and effectiveness. Automated data collection and advanced analysis makes this possible.

2.3 Workflow

Preamble: I expand on the stages I have seen, not on those I have not seen. I am part of this project; I mention what I have done ...

2.3.1 Structure of a Deployable Script



Most scripts that run on the Zensor platform have a very common structure. For a given time window, they:

1. Load some data (either raw or from InfluxDB).
2. Process it in some (clever!) way.
3. Write the results out to InfluxDB, to be shown in a dashboard.

What time window they operate on will depend on what the task is, but also on whether the script is being invoked automatically by cron, or manually. If a script is being invoked manually, this is usually to run it over historical data e.g. rerunning a script for the month of February 2020. We typically call this **backfilling**. Typically, if the script is running in cron, it's loading "recent" data, e.g. from the past hour or past day, ending at the time the script started. Scripts on the Zensor platform need to support running in both modes, so there are a few guidelines to keep in mind when writing a script.

Chapter 3

Tools employed

The idea of this chapter is to show the main tools used during the work as seen at [2.3.1], and to highlight the important pieces.

3.1 Pandas

Pandas is a data manipulation and analysis software library created for the Python programming language. It provides data structures and functions for manipulating numerical tables and time series, and it is free software distributed under the BSD three-clause licence [**Misc:OpenLDAP_license:oldap-2.7**]. The name derives from the word "panel data", which is an econometrics term for data sets that comprise observations for the same individuals over several time periods and, at the same time, is a parody of the term "Python data analysis" [**mckinney_data_2010**].

3.1.1 Series and DataFrame

Pandas is primarily used to analyse data. It supports data import from a variety of file formats, including comma-separated values (CSV), JSON, SQL database tables or queries, and Microsoft Excel [**Misc:pandas_docs**]. Further more Pandas supports a variety of data manipulation operations such as merging, reshaping, and selecting, as well as data cleaning and handling. To accomplish this, Pandas define and makes use of two important software *Classes*, Series and DataFrame, which we will now briefly introduce.

Series is a one-dimensional labeled array capable of holding any data type (integers, strings, floating point numbers, Python objects, etc.). The axis labels are collectively referred to as the *index*.

DataFrame is a 2-dimensional labeled data structure with columns of potentially different types; it some ways it is like a spreadsheet or SQL table

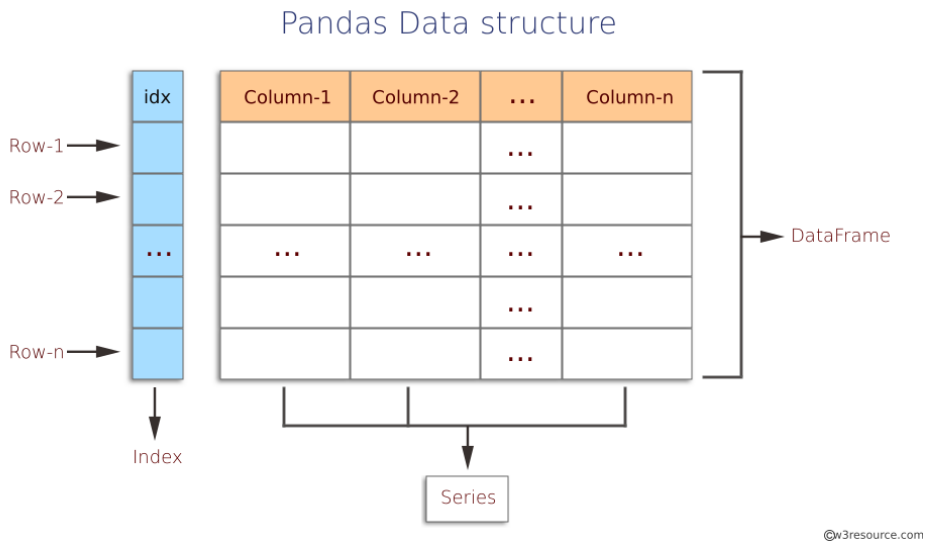


Figure 3.1: Pandas Data structure

(see figure: 3.1) and each individual column is a *Series*. It is generally the most commonly used pandas object, since it accepts many different kinds of input at makes him very flexible [**reback_pandas-dev/pandas:_2022**]. Along with the data, one can optionally pass index (row labels) and columns (column labels) arguments. By doing so you are guaranteeing the index and/or columns of the resulting DataFrame. If axis labels are not passed, they will be constructed from the input data based on common sense rules. And this is just one way of building a dataframe, a foretaste of the flexibility of this library.

3.1.2 Core Features

The idea of this subsection is to give an outline of how many possible use-case this library can cover, and, at the same time, explore a couple of them that proved to be crucial during my internship experience; let's start with the idea of *grouping*.

Groupby The name **GroupBy** should be quite familiar to those who have used a SQL-based tool or worked with relation database. This "engine" allows split-apply-combine operations on heterogeneous data sets. by "group by" we are referring to a process involving one or more of the following steps:

1. Splitting the data into groups based on some criteria.
2. Applying a function to each group independently.
3. Combining the results into a data structure

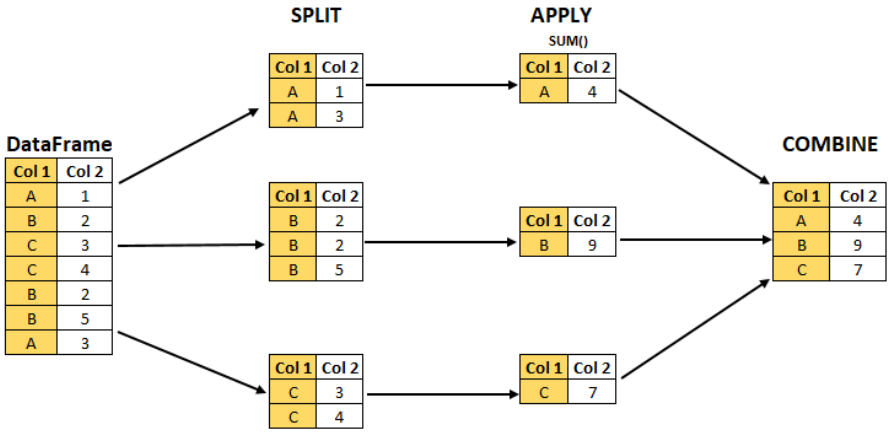


Figure 3.2: Shows the Split-Apply-Combine using an aggregation function (source: Analytics Vidhya [**Misc:pandey_split-apply-combine**])

Out of these three, the *split* step is the most straightforward. In fact, in many situations, we would like to split the data set into groups and do something with those groups [**reback_pandas-dev/pandas:_2022**]. In the *apply and combine* step, we might wish to do one of the following:

- Aggregation: compute a summary statistic (or statistics) for each group, some examples:
  - Compute group sums or means.
  - Compute group sizes / counts.
- Transformation: perform some group-specific computations and return a like-indexed object, for instance:
  - Standardize data (zscore) within a group.
  - Filling NAs (value that are not valid) within groups with a value derived from each group.
- Filtration: discard some groups, according to a group-wise computation that evaluates True or False, like:
  - Discard data that belongs to groups with only a few members.
  - Filter out data based on the group sum or mean.
- Some combination of the above: **GroupBy** will examine the results of the *apply* step and try to return a sensibly *combined* result if it doesn't fit into either of the above two categories.

Since the set of object instance methods on pandas data structures are generally rich and expressive, we often simply want to invoke, say, a DataFrame function on each group. With this engine you can try multiple different approaches, testing what is suits more your necessities, even though often is hard to define this three separates step for badly shaped data.

Time series resampling Pandas contains extensive capabilities and features for working with time series data for all domains. Using the **NumPy** datetimes dtypes, it has consolidated a large number of features from other Python libraries (like **scikits.timeseries**) as well as created a tremendous amount of new functionality for manipulating time series data. As an example, Pandas supports:

- Parsing time series information from various sources and formats
- Manipulating and converting date times with timezone information
- Moving window statistics and linear regressions

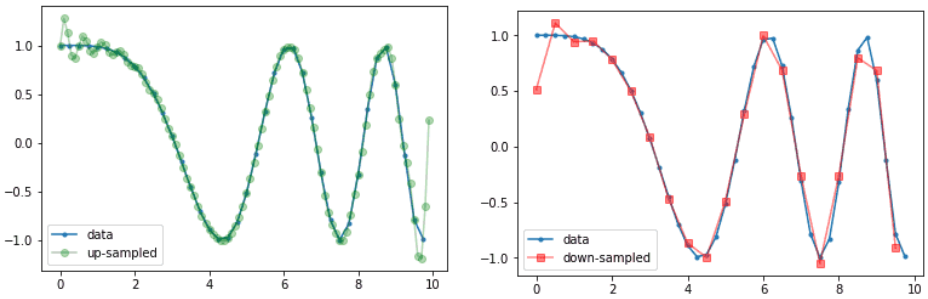


Figure 3.3: Examples of signal waveform data processed by resample

Let's now focus a bit about one key aspect of handling/manipulating sensor data, crucial in our context, the necessity of resampling. Fortunately, Pandas has, once again, a simple-to-use, powerful, and efficient functionality for performing resampling operations during frequency conversion (e.g., converting secondly data into 5-minutely data). This is also extremely common in, but not limited to, financial applications.

To keep things simple we could say that resample is a time-based **Groupby** followed by a reduction method on each of its groups; as a positive side, this method can be used directly from *DataFrameGroupBy* objects that we discussed in paragraph [3.1.2]. The resample function is very flexible and allows you to specify many different parameters to control the frequency conversion and resampling operation, both upsampling and downsampling.

Others functionality Furthermore, other time series features are available, and not only that notably:

- Date range generation and frequency conversions
- Data alignment, shifting and lagging
- Integrated missing data handling
- Data set reshaping, pivoting, merging and combining
- Label-based slicing, sophisticated indexing, and big data set sub-setting
- Insertion and deletion of columns in a data structure.

Conlcusion Pandas provides a solid foundation upon which a very powerful data analysis ecosystem can be established, especially since the library is performance-optimized, with important code paths implemented in Cython or C.

3.2 InfluxDB

Time Series In mathematics, a time series is a series of data points which are indexed (or listed or graphed) in time order (see 3.4). More generally, a time series is a sequence taken at evenly spaced intervals over a period of time. As a result, it's a succession of discrete-time data. Ocean tidal heights, sunspot counts, and the Dow Jones Industrial Average's daily closing value are all examples of time series.

3.2.1 TSDB: time series database

It follows that a time series database (TSDB) is a software system that is designed to store and serve this peculiar type of data, time series, using time(s) and value(s) pairs(s). Timescale (popular Time Series Database (TSDB)) CEO *Ajay Kulkarni* [**Misc:asay_why_time_series**] put it:

> [T]ime-series datasets track changes to the overall system as INSERTs, not UPDATEs.
>
> This practice of recording each and every change to the system as a new, different row is what makes time-series data so powerful. It allows us to measure change: analyse how something changed in the past, monitor how something is changing in the present, predict how it may change in the future.
>
> [So] here's how I like to define time-series data: data that collectively represents how a system/process/behaviour changes over time.

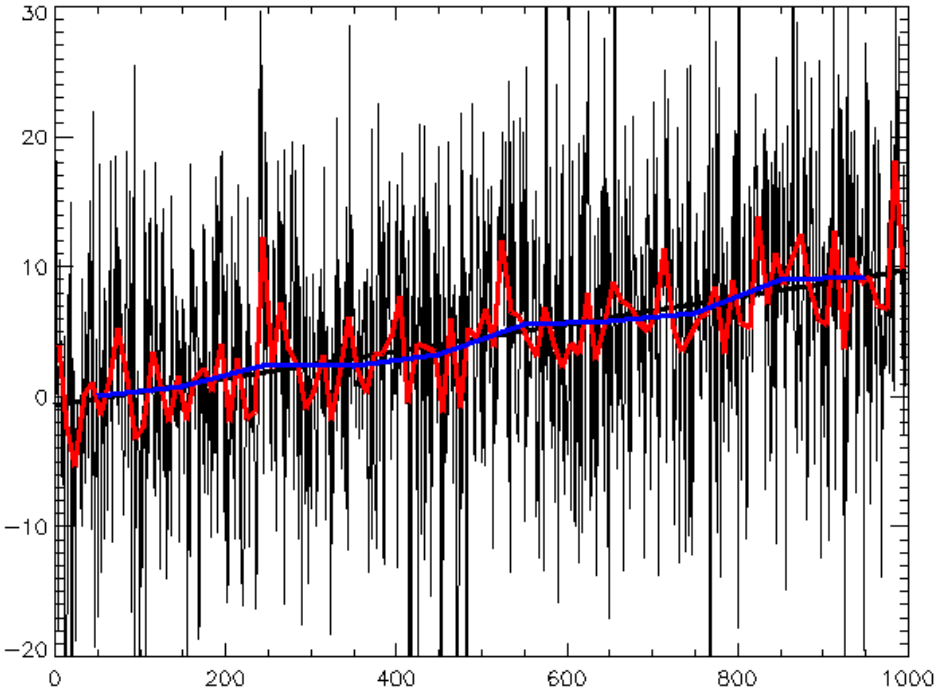


Figure 3.4: Random data plus trend, with best-fit line and different smoothings applied. (source: Wikimedia Commons [**file:random-data-plus-trend-r2**])

Although it is possible to store time-series data in many diverse database types, the design of these systems with **time** as a **key** index is distinctly different from relational databases, which reduce discrete relationships through referential models. In many cases, the repositories of time-series data will utilize compression algorithms to manage the data efficiently [**Book:devops_cookbook**]. Furthermore, time series databases can also be configured to regularly delete old data, unlike traditional databases which are designed to store data indefinitely.

3.2.2 Influx solution

InfluxDB is an open-source time series database *(TSDB)* created by the InfluxData organization [**Misc:influxdata_website**]. It is written in the Go programming language and is used for time series data storage and retrieval in various sectors such as operations monitoring, application metrics, Internet of Things, and, especially important in our context, sensor data and real-time analytics. It also supports the processing of data from Graphite, a data logging and graphing tool for time series data. **1.Quest'ultima frase non è così rilevante**

Core Features InfluxDB has no external dependencies and provides a SQL-like vocabulary with built-in time-centric functions for querying a data structure made up of measurements, series, and points, which listens on port 8086 [**Misc:influx_docs**].

time	location	scientist	butterflies	honeybees
2015-08-18T00:00:00Z	1	langstroth	12	23
2015-08-18T00:00:00Z	1	perpetua	1	30
2015-08-18T00:06:00Z	1	langstroth	11	28
2015-08-18T05:54:00Z	2	langstroth	2	11
2015-08-18T06:00:00Z	2	langstroth	1	10
2015-08-18T06:06:00Z	2	perpetua	8	23
2015-08-18T06:12:00Z	2	perpetua	7	22

Table 3.1: Sample time series dataset: number of butterflies and honeybees counted by two scientists

Each point is made up of a fieldset and a timestamp, which are key-value pairs. These form a series when they are grouped together by a set of key-value pairs known as a tagset. Finally, a measurement is created by grouping series together using a string identification. 64-bit integers, 64-bit floating points, strings, and booleans are among the possible values as shown in the table above [3.1]. The time and tagset are used to sort the points. As a side note it is important to know that data is downsampled and removed according to retention policies, which are set by measurement and that *Continuous Queries* are executed on a regular basis and the results are stored in a goal measurement.

Design Tradesoff InfluxDB is a time series database and optimizing for this use case involves a number of trade-offs, primarily to increase performance at the cost of functionality [**Misc:influx_docs**]. Here is a list of three design ideas that lead to compromises that I personally experienced during my experience:

1. Deletes are a rare occurrence. When they do occur, it is almost always against large ranges of old data that are cold for writes.
2. Updates to existing data are a rare occurrence, and contentious updates never happen. Time series data is predominantly new data that is never updated.
3. Many time series are ephemeral. There are often time series that appear only for a few hours and then go away, e.g., a new host that gets started and reports for a while and then gets shut down.

2.Potrebbero essere più di 3, mi sembrava un buon compromesso

Pros	Cons
Restricting access to deletes and updates allows for increased query and write performance	Delete and Update functionality is significantly restricted, since influxDB is not CRUD
InfluxDB is good at managing discontinuous data	Schema-less design means that some database functions are not supported e.g. there are no cross table joins

Table 3.2: Pros and cons of InfluxDB

Conclusion It is therefore no surprise to conclude that, when compared to a general purpose relational database like SQL Server, InfluxDB, using default single node configuration, outperformed both write speed, disk storage usage (by a factor of 27x) and query execution time, where InfluxDB is up to 20x faster with an average of 8x faster [**Misc:noor_2017_universit**]. It can be seen that InfluxDB, tailored-made for Time Series data, is released after the other competitive technologies (like Graphite, TimescaleDB, Prometeus) and yet still among the top list. `https://db-engines.com/en/ranking_trend/time+series+dbms` **3.Non so se valga la pena citarlo** The SQL-like query language helps it make easier to use and adapt by people who are use to working with relational databases like MySQL.

3.3 Grafana

Grafana is a web-based analytics and interactive visualization application that runs on a variety of platforms. When connected to supported data sources, it produces web-based charts, graphs, and alerts. Grafana Enterprise, a paid version with more features, is available as a self-hosted installation or as a Grafana Labs cloud service account [**Misc:grafana_labs_website**]. Grafana is split into two parts: a front end and a back end, both of which are built in TypeScript and Go and, through a plug-in, system, it can be expanded, adding specific customizations to the existing platform [**Misc:grafana_docs**]. Using interactive query builders, end users can develop complicated monitoring dashboards. But what is a dashboard anyway?

3.3.1 Dashboard what is it?

A dashboard, in business, is a type of graphical user interface that often enables quick access to key performance indicators (KPIs) related to a certain

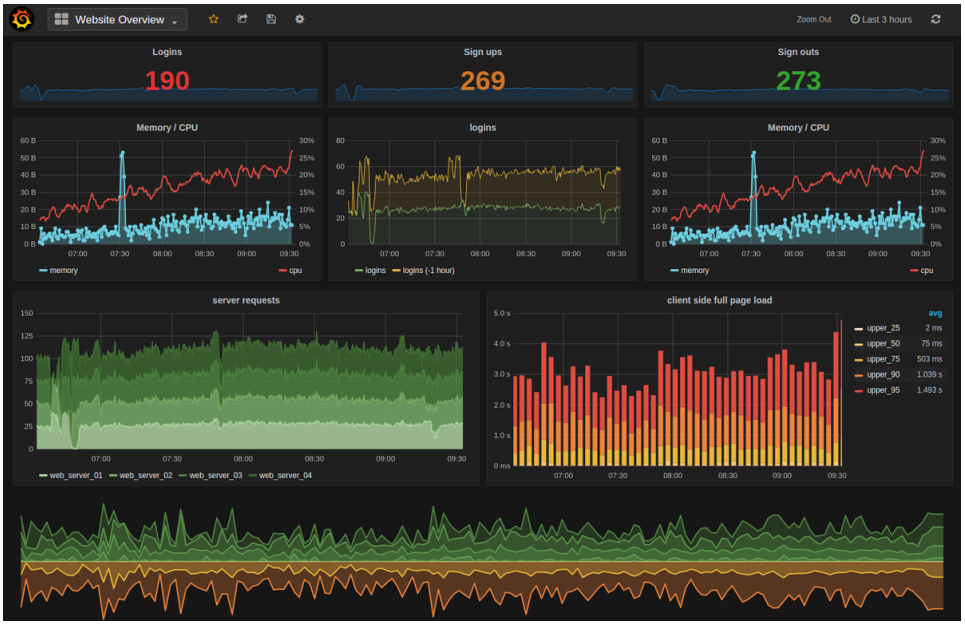


Figure 3.5: Grafana Graph Visualization (Source: Flickr [**file:screenshots_grafana**])

goal or business activity. In our context, ”dashboard” refers to a ”progress report” or ”report” and, as a type of data visualization, it is mostly accessible by a web browser and is usually linked to regularly updating data sources. The term dashboard is derived from the automotive dashboard, where drivers may monitor the primary functions at a glance using the instrument panel.

The success of dashboard projects depends on the relevancy/importance of information provided within the dashboard. This includes the metrics chosen to monitor and the timeliness of the data forming those metrics; data must be up-to-date and accurate. Well known dashboards include Google Analytics dashboards, used on 55% of all websites [**Misch:w3techs_usage**], which show user activity on a website, or the UK government, and similar for each country, coronavirus tracker, for the COVID-19 pandemic [**Misc:uk_covid_dash**].

An interesting project is the GLAM Wiki dashboard, from Israel [**Misc:wikidata_glam_project**]. Its purpose is to assist GLAM institutions (galleries, libraries, archives, and museums) in tracking the use of their free-content files that they have submitted to Wikimedia projects. Based on multiple specified indices and several time frames, the dashboard visualizes statistical data that shows the extent of exposure and usage of these public-domain assets. The collecting data, which is presented in a variety of diagrams and graphs, allows the institutions to obtain insights, discover patterns and preferences, and understand the overall impact of these free materials on the global audience of Wikimedia-project users.

3.3.2 Key strengths

Now that we are familiar with the "dashboard" concept, we can highlight why we should use Grafana; here are some of the key features [**Article:comprehensive_study_g**

- **Visualize** fast and flexible visualization with a variety of options allows data to be displayed the way the user wants it;
- **Dynamic Dashboard** dynamic and reusable dashboards may be created using template variables.
- **Explore Metric** ad-hoc queries and dynamic drill-down permits for data discovery. View splits and side-by-side comparisons of different time ranges and data sources is easily achievable;
- **Explore Logs** fast switch from metrics to logs preserving label filters. Furthermore, searching the logs is rather quick and can be performed on live streams;
- **Alerting** most of the vital/operational metrics may be alerted visually and different types of notifications (SMS, mail, Slack) may be despatched with the aid of Grafana;
- **Mixed Data Source** the same chart can have different data source: these can be selected based on queries, with built-in support for most of the prominent data sources available in the market as well as custom ones;
- **Annotations** graphs may have events that can be annotated, two solutions are possible: use native annotation store, with the ability to add annotation events directly from the graph panel or via the HTTP API, or querying other data sources. Event metadata and tags can be seen when hovering over events.

Loading speed of Grafana dashboards **4.Potenzialmente da rimuovere/espandere** Loading speed of a Grafana dashboard depends on 5 major things:

1. pre-selected and saved time window: the larger the time period you query, the longer it takes to open and display the contents;
2. data frequency in the panels: in case of the very high frequency, non aggregated data, even if selected time period is minutes, it will take time to load;
3. the number of panels with the data inside;
4. your database structure;
5. whether calculations have to happen inside the panel before the data is displayed.

Conclusion Grafana is the right choice when visualizing infrastructure, applications, network devices, sensors, and more. This is a great 24/7 monitoring solution for NOC and DevOps teams. It can also help to manage all data from other application monitoring tools like AppDynamics, New Relic, Splunk, Dynatrace and all-in-one web interface for data viewing, alerting, and reporting. A further comparison with other data visualization tools, such as Power BI and Tableau, might make interesting reading.