Interactive Visualization for Time Series analysis

Xavier Charef
Rémi Sanchez
Thai-An Than Trong
Christophe Thibault

State of the Art

Safran Group

MS Big Data Télécom ParisTech
Projet Fil Rouge

Contents

Abstract

This document describes the State of Art of time-series data visualization, and solution of an implementation of a powerful tool for multivariate time-series visualization too.

1 Introduction

1.1 Data visualization

"A picture is worth 1000 words" (proverb)

"A picture can also be worth 1000 data points." D3.js website.

Data Visualization is the representation of data in a pictorial or graphical format. It enables decision makers to see analytics presented visually, for them to discover or identify new patterns. Data visualization can roughly be categorized into two applications:

- Exploration : In the exploration phase, the data analyst will use many graphics that are mostly unsuitable for presentation purposes yet may reveal very interesting and important features. The amount of interaction needed during exploration is very high. Plots must be created and modifications like sorting or rescaling should happen instantaneously so as not to interrupt the line of thought of the analyst.
- Presentation : Once the key findings in a data set have been explored, the findings must be presented to a broader audience interested in the data set. These graphics often cannot be interactive but must be suitable for printed reproduction. Furthermore, some of the graphics for high-dimensional data are all but trivial to read without prior training, and thus probably not well suited for presentation purposes - especially if the audience is not well trained in statistics.

Visual representation have undoubtedly a massive impact on communicating facts and information throughout history. However, it only became a self-contained research field about twenty years ago. In 1987 the notion of visualization in scientific computing was introduced by McCormick et al. (1987).

> "Visualization is a method of computing. It transforms the symbolic into the geometric, enabling researchers to observe their simulations and computations. Visualization offers a method for seeing the unseen. It enriches the process of scientific discovery and fosters profound and unexpected insights."

The purpose of this new field of research is to combine the capabilities of human visual perception with the efficient processing power of computers to support users in analyzing, understanding, and communicating their models and concepts based on data.

Furthermore, the exponential growth of data make data visualization even more essential to identify patterns and find significations that are not visible if we just look at raw data.

Given the complexity of data, in order to provide a meaningful solution, we combine insights from many fields : data mining, statistics, graphic design, and information visualization. Together These fields create a whole process to generate an exhaustive, efficient and autonomous visualization. According to Ben Fry in his book Visualizing Data[1], the steps are the following :

1. Acquire : Obtain the data from a file, a disk or a source over a network.
2. Parse : In order to manipulate the large amount of data, it is necessary to provide some structure for the data's meaning, and order it into categories.
3. Filter : Only keep interesting data.
4. Mine : Apply methods from statistics or data mining to discover patterns.
5. Represent : Choose a visual model.
6. Refine : Improve the previous representation in order to make it clearer and more visually engaging.
7. Interact : Add methods to manipulate data or control which features to visualize. Interactive visualization enables us to see the data with different angles.

For our project we will need to pay attention to these steps to make sure our visualization is understandable.

1.2 Time Series

Space and time are two distinct dimensions. While space can in principle be navigated arbitrarily in all three spatial dimensions, and it can easily be reversed to go back to where we came from, time does not allow active navigation. We are indeed constrained to the unidirectional character of time. We cannot go back to the past. Moreover, humans do not have sense for perceiving time directly, making time visualization particularly challenging.

Not only does time have a particular hierarchical structure of granularities, as for example minutes, hours, days, weeks, it also has different forms of divisions : 60 minutes correspond to one hour, while 24 hours make up one day. Time also has natural cycles and re-occurrences, such as seasons, but also social cycles such as holidays or school breaks. All these characteristics lead to the necessity of treating time-oriented data, i.e., data that are inherently linked to time, with appropriate visual and analytical methods to explore and analyze them.

The analysis of experimental data that have been observed at different moments in time leads to new and unique problems in statistical modeling and

[1] `http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.173.2453&rep=rep1&type=pdf`

inference. However, many visualizations techniques treat time just as a numeric parameter among other quantitative dimensions, instead of exploiting time's special character.

Times series are an ordered sequence of values of a variables at equally spaced time intervals. Accounting for the special characteristics can be beneficial from a data modeling point of view, since we can use different calendars and granularities according to the application domain (e.g., fiscal quarters or academic semesters and then choose between months, days, hours, seconds), thus enabling value aggregation along granularities.

Time series are used for many applications such as economic forecasting, sales forecasting, process and quality control ... The broad range of applications lies on the importance of understanding the underlying forces and structure that produced the observed data, as well as the possibility of forecasting, monitoring or even feedback and feed-forward control.

2 Data visualization for Time Series

2.1 Graphs for Time Series Visualization

2.1.1 Parallel Coordinates

Parallel plots escape the dimensionality of two or three dimensions and can accommodate many variables by plotting the coordinate axes in parallel. Ge-

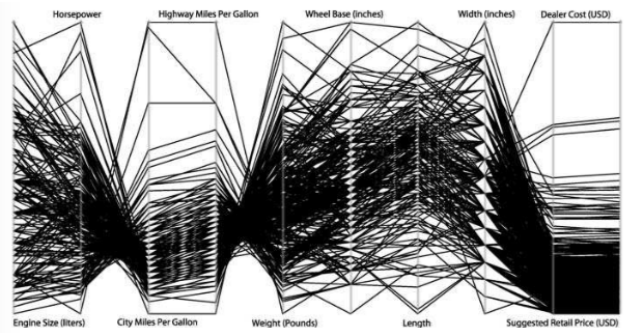


Figure 1: Example of parallel coordinates plot for some variables on almost 400 cars

ometries properties of parallel coordinates plots allow visualization properties of high-dimensional data-mining and classification methods. The most interesting aspects in using parallel coordinates plots are the investigation of groups/clusters, outliers, and structures over many variables at a time. The main uses of parallel coordinates plots are:

- Overview: an ideal tool to get first overview of a data set (lots of variables, and observations - relations between variables).
- Profiles: can be used to visualize the profile of a single case via highlighting. Profiles are not only not restricted to single cases but they can be

plotted for a whole group, to compare the profile of that group with the rest of the data. Using parallel coordinates plots to profile is especially efficient when the coordinate axes have an order like time.

- Monitor: Working on subsets of a data set parallel coordinates plots can help link features of a specific subset to the rest of the data set. Parallel coordinates plots can help find the major axes that influence the configuration of the multidimensional scaling.

Limits: Parallel coordinates are often overrated with respect to the insight they provide into multivariate features of a data set. Scatter-plots are superior for investigation 2-D features, but scatter-plots matrices need more space to plot the same information than Parallel Coordinates. Detection of multivariate outliers is not something that can usually be directly aided by parallel coordinates. Detecting features in parallel coordinates that are not visible in a 1-D or 2-D plot is rare. Parallel coordinates are extremely useful for interpreting the findings of multivariate procedures like outlier detection, clustering, or classification in a highly multivariate context.
Solution against over-plotting is α-blending (with α, percent. of opacity).

Sorting and scalings: As we wrote before, parallel coordinate plots are especially useful for variables with an order as time. In this case, scaling and sorting issues are very important for a successful exploration of the data set. Sorting in parallel coordinate plots is crucial for the interpretation of the plots, as interesting patterns are usually revealed at neighboring variables. Variables may share the same scale, because it is more helpful to be able to sort the variables according to some criterion. This can be statistics of the variables like minimum, mean, range, or standard deviation or the result of a multivariate procedure. Sorting axes can reduce the visual mess of a parallel coordinate plot substantially.
Besides the default scaling, which consists in plotting all values over the full range of each axis between the minimum and the maximum of the variable, there are many other useful scalings. There are two types of scaling options, the ones that have an individual scale for each axis and the ones that use a common scale over all axes (implying that all the axis are in the same unit of measurement). We can also scale data by specifying the alignment of the values, which can be aligned at :

- mean
- median
- specific case
- specific value

2.1.2 Radar charts (Kiviat charts) - Spider Charts

Radar chart is also known as spider chart or star plot because it looks like spider's web or stars. Radar chart is a graphical method of displaying multivariate data in the form of a two-dimensional chart of three or more quantitative and

qualitative variables represented on axes starting from the same point : origin. The radar chart is a visual representation as a chart consists of a sequence of equi-angular spokes, called *radii*, with each spoke representing one of the variables. The data length of a spoke is proportional to the magnitude of the variable for the data point relative to the maximum magnitude of the variable across all data points. A line is drawn connecting the data values for each spoke. This gives the plot a star-like appearance and the origin of one of the popular names for this chart.
This makes them useful for seeing which variables have similar values or if there are any outliers amongst each variable. Radar Charts are also useful for seeing which variables are scoring high or low within a dataset, making them ideal for displaying performance. Radar charts are especially good for visualizing com-

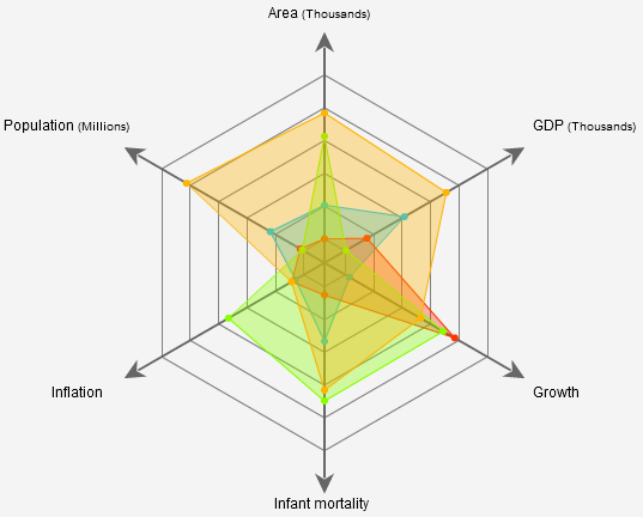


Figure 2: Example of Radar Chart representation

parisons of quality data. Many attributes can be easily compared each along their own axis, and overall differences are apparent by the size and shape of the polygons. Another advantage of the radar chart is that many variables can be represented next to each other while still giving each variable the same resolution. The comparison of observations on a radar chart can become confusing once there are more than a couple webs on the chart, or if there are too many variables, and therefore too many axes, crowding the data. This problem can be fixed by lowering the opacity of the polygons, but as more polygons are layered on top of each other, this could make distinguishing colors and individual polygons difficult.

2.1.3 Time Wheel

Idea behind the Time-Wheel method is to present the time axis at the center of visualization and to arrange the other variables around it and to assign each axis a unique color. The points of each axis are connected to the time axes to show the relation between the axes.
The two parallel axes are the easiest ones to interpret and the two perpendicular

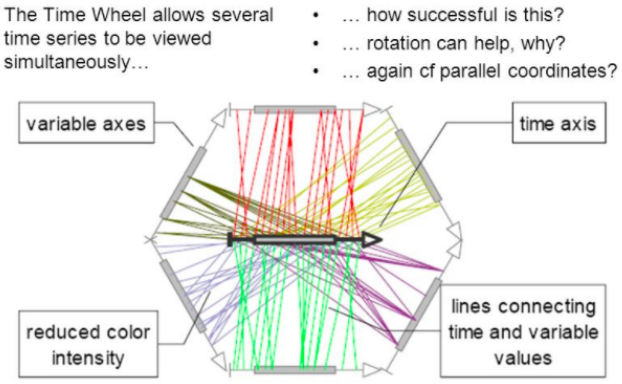


Figure 3: Exemple and explanation of Time-Wheel representation (réf: Tominski)

ones the hardest. The Time-Wheel allows an interactive rotating of the axes. Further techniques such as colour fading and axes length adjustment are used to avoid over-clogging of the visualisation [Tominski].

2.1.4 Grand tour visualization and projection pursuit

The Grand Tour is a highly exploratory tool. Basically the Grand Tour examines a series of scatter-plots obtained by a smoothly changing sequence of projection methods. With the Grand Tour, one can dynamically explore the structure of multi-dimensional data projected onto a two dimensional window. Whereas in classical parallel coordinate plots the data are still projected to orthogonal axes, the Grand Tour permits one to look at arbitrary non-orthogonal projections of the data, which can reveal features invisible in orthogonal projections.

The default method for choosing the new projections to view is to use a random

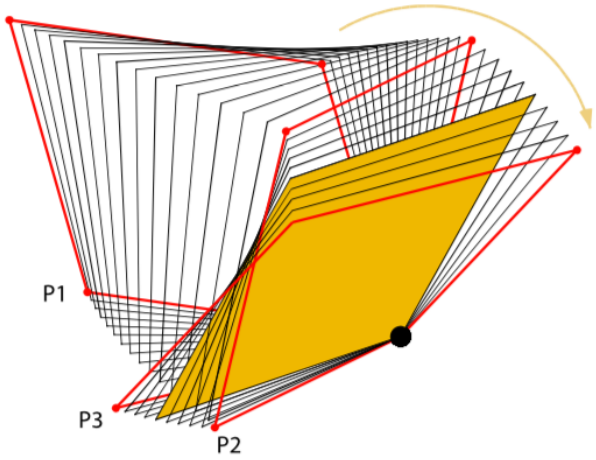


Figure 4: Example path of a Grand Tour

sequence. The sequence is determined by randomly selecting a frame describing a target plane from the space of all possible projections. A grand tour may be considered to be an interpolated random walk over the space of all planes. A target frame is chosen randomly by standardizing a random vector from a standard multivariate normal distribution. The Grand Tour basically examines a series of scatter-plots obtained by a smoothly changing sequence of projection methods. With the Grand Tour one can dynamically explore the structure of multi-dimensional data projected onto a two dimensional window.

Figure 5 is from an article about Flexible Linked Axis[2] that consists to create an easily axes structure with the aim that user himself could select the suitable solution.

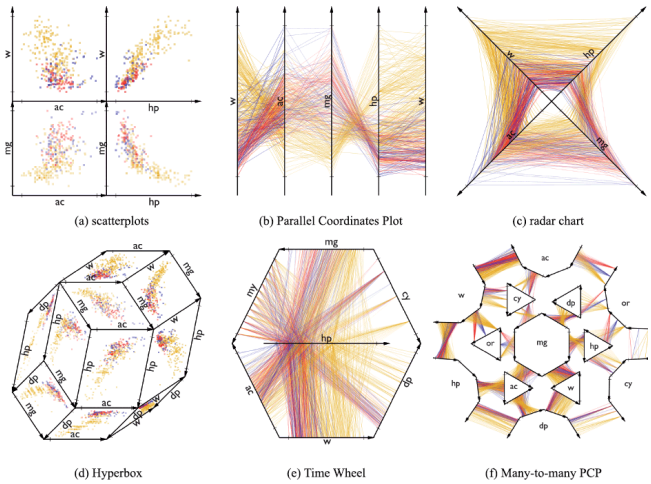


Figure 5: Models' representation: Scatter Plot(a), Parallel Coordinates Plot(b), Radar Chart (c), Hyperbox (d), Time Wheel (e) and Many-to-many Parallel Coordinates Plot (f)

2.2 Original Time Series Visualizations

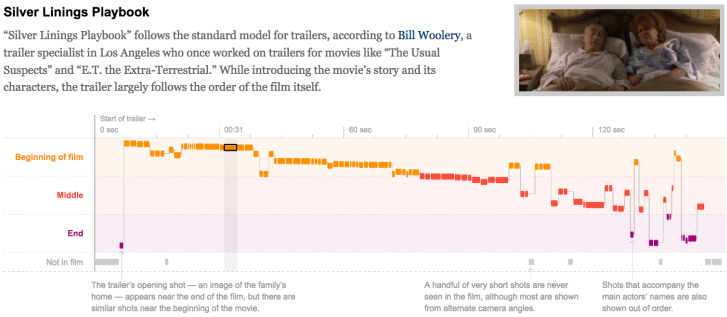
Data visualizations are very common in medias, and websites. They enable to show information based on data with interactive tools, thus replacing dense and hardly understandable texts. Here are some examples of complex and original time series visualizations extracted from articles and websites.

"Dissecting a Trailer: The Parts of the Film That Make the Cut" The following data visualization comes from a New York Times article [3] and describes the composition of the trailer of the movie Silver Linings Playbook. The horizontal axis represents the time, scaled on the duration of the trailer ; the left part corresponds to the beginning of the trailer, while the right part corresponds to the end. The vertical axis is divided in three sections : the

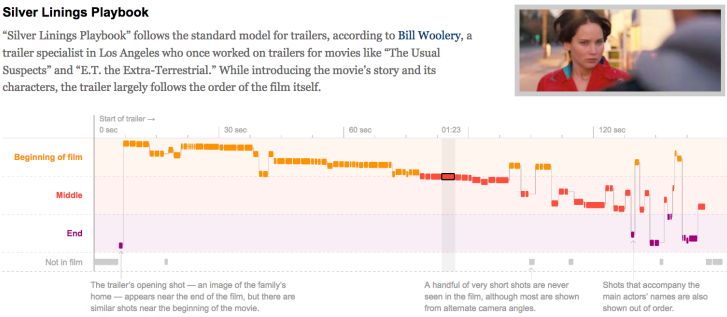
[2]Flexible Linked Axes for Multivariate Data Visualization

[3]`http://www.nytimes.com/interactive/2013/02/19/movies/awardsseason/oscar-trailers.html`

beginning of the movie, the middle, and the end. Each part of the trailer is decomposed and associated to the matching moment of the movie. At the end, if we take a global look at the trailer, we can see that it follows the order of the actual movie for the most part.



(a) First view

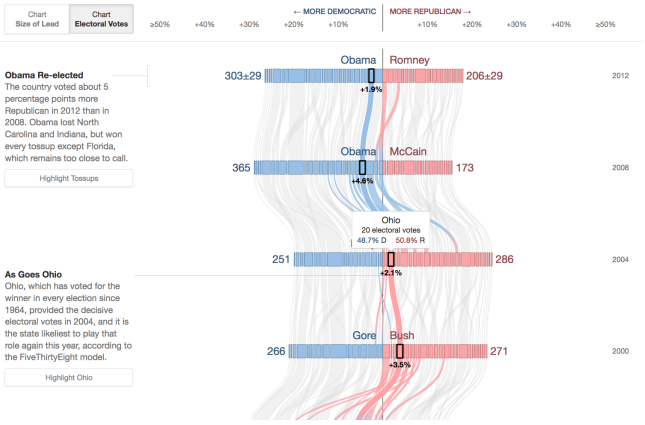


(b) Second view

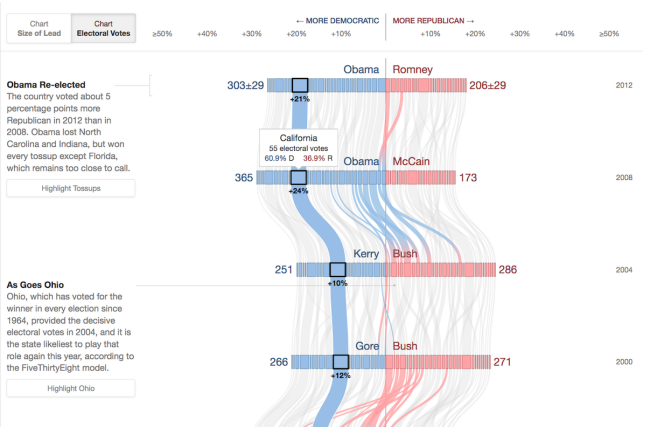
Figure 6: Time series visualization with the movie Silver Linings Playbook. The visualization shows the composition of the trailer (horizontal axis) based on the extracts from the entire movie, which is split in three categories: beginning, middle, and end (vertical axis). As many movies, the trailer largely follows the order of the film itself.

"Over the Decades, How States Have Shifted" The following data visualization also comes from a New York Times article [4] that was published on November 12th, just after the US elections. Its purpose is to show political trends for each state, and whether the majority is Republican or Democrat. It mainly highlights the influence of swing states in the presidential elections results along the years in the American history. The data visualization is based on complex bar charts divided into the different states, and the time data is recreated with the graph lines. This way, we can recognize states that used to vote a given party, and then change their mind afterwards in other elections.

[4] `http://www.nytimes.com/interactive/2012/10/15/us/politics/swing-history.html`



(a) Example of a swing state



(b) Example of a safe state

Figure 7: Time series visualization for the presidential elections in the United States of America. It retraces the evolution of states majorities throughout the years of elections and the impact of swing states on the presidential results.

Janet L. Yellen, on the Economy's Twists and Turns The illustration below come from a New York Times article[5] that displays the evolution of inflation rate depending on the unemployment rate from 1969 to 2013.

Horizon graphs Horizon graphs enable to visualize many time-dependent variables in a limited space. They could also be used as a first tool to explore data and find interesting areas to investigate further using more precise graphs. We can see an example of this for the evolution of the price of different items[6].

Time curves Time curves represent temporal data in 2D by defining a similarity function that gives for two points the similarity between them (according to all the variables). Then we plot points on a graph for each measure in time

[5] `http://www.nytimes.com/interactive/2013/10/09/us/yellen-fed-chart.html`
[6] `https://flowingdata.com/2015/07/02/changing-price-of-food-items-and-horizon-graphs/`

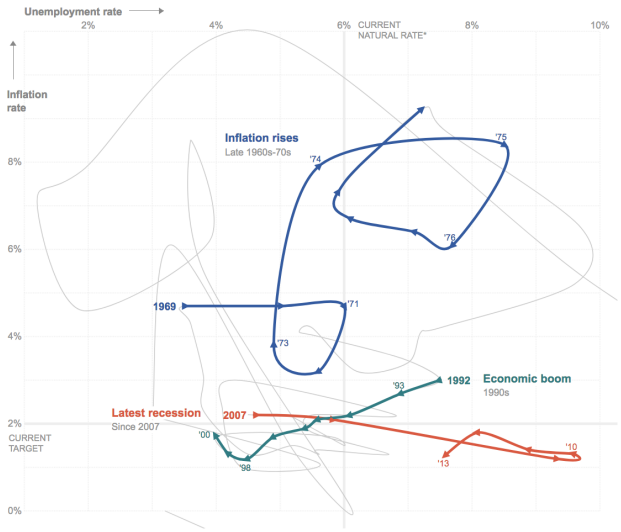


Figure 8: Time series visualization for the inflation rate evolution throughout the years from 1969 to 2013

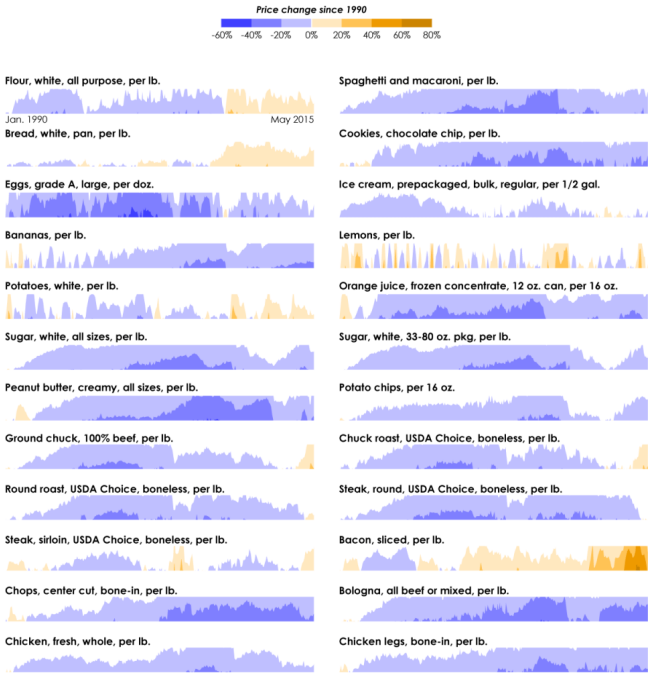


Figure 9: Visualization of the evolution of price of different items between 1990 and 2015

connected by a line which is representing the time (t-1 linked to t linked to t+1) and we place the point according to it's similarity with the previous points.[7].

[7] `http://ieeexplore.ieee.org/document/7192639/`

Kiviat tubes The idea behind the Kiviat tubes is to represent an instant in time by a kiviat graph and then to put them next to each other along an axis representing the time. It's a 3D representation.

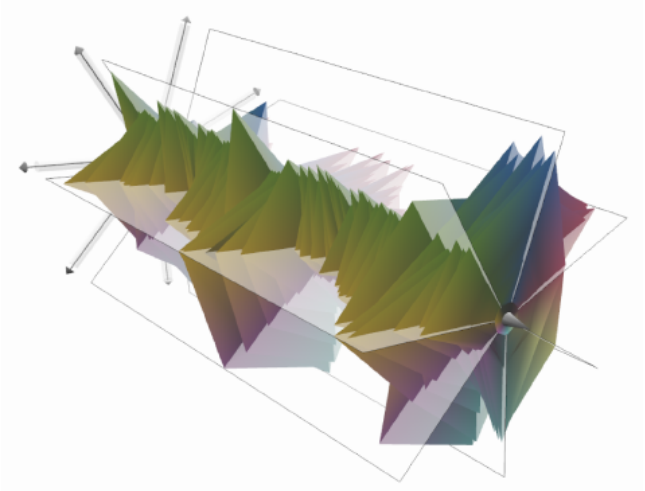


Figure 10: Example of kiviat Tube

Stacked graphs It's a classical representation of multi-variable time series where each graph for a variable is stacked on top of one another.

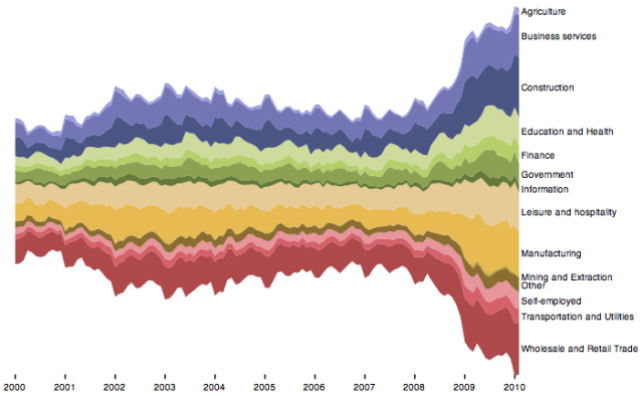


Figure 11: Example of stacked graph

3 Existing tools for data visualization

In this part, we will present some commercial tools that are dedicated to data visualization.

3.1 Existing softwares for Time Series visualization

3.1.1 General Data Visualization Softwares with Time Series functionalities

- **Tableau** Tableau Software's built-in date and time functions with drag and drop enable to analyze time trends. It can also drill down with a click, analyze times by day of the week, and easily perform time comparisons like year-over-year growth and moving averages. No native chart type in Tableau. It can integrate a broader range of data sources including spreadsheets, CSV, SQL databases, Salesforce, Cloudera Hadoop, Firebird, Google Analytics, Google BigQuery, Hortonworks Hadoop, HP Vertica, MS SQL Server, MySQL, OData, Oracle, Pivotal Greenplum, PostgreSQL, Salesforce, Teradata, and Windows Azure Marketplace. It can also connect with R that powers the analytical capabilities of the tool. It can finally connect with *Big data sources*.

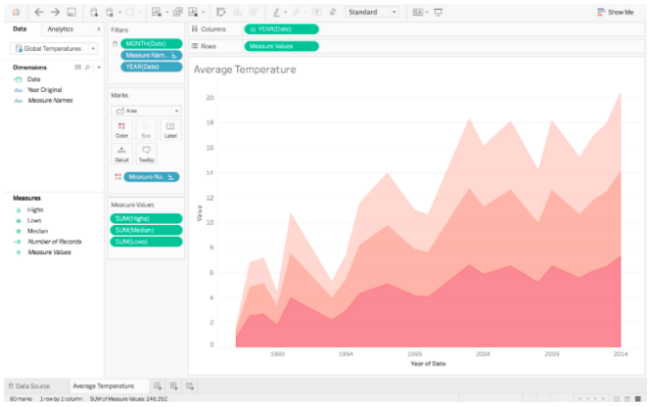


Figure 12: Example of view of Tableau Software

- **QlickView** easy to use to explore the hidden trends. Its actively engaged community and resources help to learn this software in the best possible manner. It integrates with a very broad range of data sources like Amazon Vectorwise, EC2, and Redshift, Cloudera Hadoop and Impala, CSV, DatStax, Epicor Scala, EMC Green Plum, Hortonworks Hadoop, HP Vertica, IBM DB2, IBM Netezza, Infor Lawson, Informatica Powercenter, MicroStrategy, MS SQL Server, My SQL, ODBC, Par Accel, Sage 500, Salesforce, SAP, SAP Hana, Teradata, and many more. It can also connect with R using an API integration.

- **Spotfire** That is a data visualization tool similar to Tableau. The main difference is that it can integrate better with other TIBCO products (real time reports) and can be arguably seen as more enterprise friendly due to its scalability.

- **ElasticSearch + Kibana** Kibana[8], the data visualization engine of the ElasticSearch stack, allows to natively interact with all data in Elastic-

[8] `https://www.elastic.co/fr/products/kibana`

search via custom dashboards. Elasticsearch is an open-source, broadly-distributable, readily-scalable, enterprise-grade search engine. Accessible through an extensive and elaborate API, Elasticsearch can power extremely fast searches that support your data discovery applications.

- **Zeppelin**[9] is an analytical tool used to manipulate big data and visualize them. Web-based notebook that enables data-driven, interactive data analytics and collaborative documents with SQL, Scala and more.

3.1.2 Time Series Visualization Softwares

- **Facette**[10] Opensource solution to display time series data from various sources — such as collectd, Graphite, InfluxDB or KairosDB — on graphs. Facette contains 2 components: a back end and a front end. The back end interacts via connectors with the resources hosting the data. The front end make API calls to the back end.

- **Cluvio**[11] Cluvio is a cloud analytics platform for startups and SMEs that allows you to create dashboard and reports within minutes using SQL. Cluvio works with Redshift, Postgres, MySQL, MariaDB, AWS Aurora, Google Cloud Spanner, Google Big Query, Google CloudSQL, Microsoft SQL Server, Oracle and Exasol. It is easy to parametrize reports and make dashboards interactive with time based and custom filters. In Cluvio you can run custom R script on top of the results of a SQL query. This allows to leverage the power of R for analysis that require more effort to do in raw SQL.

- **Visualr**[12] It is a state of the art data visualization tool that provides the enterprises with an extensively flexible connectivity in terms of multiple data sources. It is a single platform for multidimensional users working securely and collaboratively.

- **NVD3.js**[13] NVD3.js is a project working d3.js.Its code is inspired by the one of Mike Bostock (co-developer of d3.js). Its library a contains many charts. Among them we find parallel coordinates, scatter charts, histogram, line charts. The aim of this library is to create reusable graphs (the graph is considered as an object(closure) with getter-setter methods that can easily modify the layout, the axes or the scales).

3.2 Existing libraries for Time Series visualization

3.2.1 General Data Visualization Libraries with Time Series functionalities

- **Plotly** Plotly's Python library is free and open source. It is an interactive, browser-based graphing library for Python. Built on top of plotly.js, plotly.py is a high-level, declarative charting library. plotly.js

[9] `https://zeppelin.apache.org/`
[10] `https://facette.io/`
[11] `https://www.cluvio.com/`
[12] `https://visualrsoftware.com/`
[13] `http://nvd3.org/`

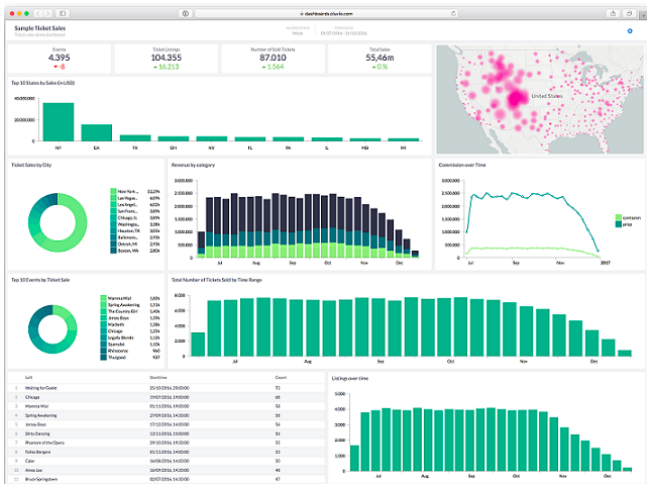


Figure 13: Example of a Cluvio dashboard

ships with over 30 chart types, including scientific charts, 3D graphs, statistical charts, SVG maps, financial charts, and more.

- **MplD3** The mpld3 project brings together Matplotlib, the Python-based graphing library, and D3js, the popular JavaScript library for creating interactive data visualizations for the web. Result is an API for exporting matplotlib graphics to HTML code which can be used within the browser, within standard web pages, or tools such as the IPython notebook.

- **d3.js** D3.js is a JavaScript library that is geared toward focusing on data and constructing data visualization projects. "D3 (Data-Driven Documents or D3.js) is a JavaScript library for visualizing data using web standards. D3 helps you bring data to life using SVG, Canvas and HTML. D3 combines powerful visualization and interaction techniques with a data-driven approach to DOM manipulation, giving you the full capabilities of modern browsers and the freedom to design the right visual interface for your data."

- **Highcharts**[14] Highcharts is great for implementations of more standard charts and graphs in a cross browser way. It is theme-able and customizable.

- **amcharts**[15] JavaScript/HTML5 charts including serial (column, bar, line, area, step line, smoothed line, candlestick and ohlc graphs), pie/donut, radar/polar and xy/scatter/bubble charts.

3.2.2 Time Series Visualization Libraries

- **MetricsGraphics** MetricsGraphics.js[16] is a library built on top of D3 that is optimized for visualizing and laying out time-series data. It pro-

[14] `https://www.highcharts.com/`
[15] `http://www.amcharts.com/javascript-charts/`
[16] `https://www.metricsgraphicsjs.org/`

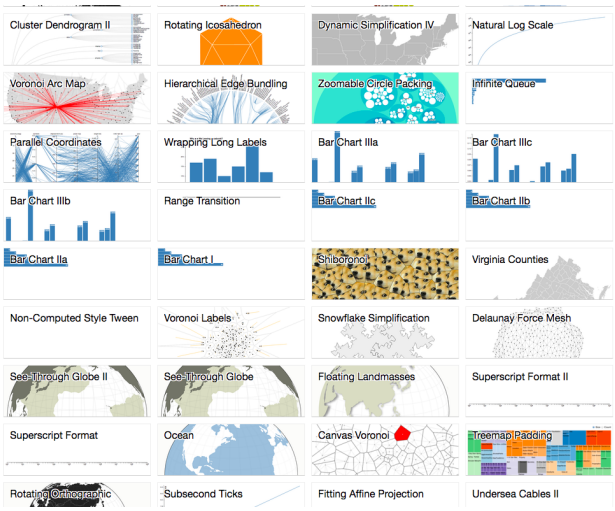


Figure 14: Example of several data visualization (https://bl.ocks.org/mbostock)

vides a simple way to produce common types of graphics in a principled, consistent and responsive way. The library currently supports line charts, scatter-plots, histograms, bar charts and data tables as well as features like rug plots and basic linear regression.

- **Dygraphs** Dygraphs[17] is an open source JavaScript library that produces interactive, zoom-able charts of time series. It is really excellent for large data sets.

- **Timeplot** Timeplot[18] is focused entirely on graphing time-series data, with support for annotating the graph with temporal events.

- **Cubism**[19] Cubism.js is a d3 plugin for visualizing time series. It can be used for real-time dashboard as Cubism.js fetches incrementally the time series data from the beginning. However, Cubism fetches only the latest values so that it reduces the server load. This brings a huge scalability to Cubism regarding the number of metrics it supports (up to hundreds). Despite asynchronous fetching, rendering is synchronized so that charts update simultaneously, further improving performance and readability.

  We can also put graphs on top another to ease comparison between them as shown in figure 16. Finally, Cubism is flexible as it supports resources like Graphite and Cube with built-in libraries or can be extended to fetch data from other sources.

[17] `http://dygraphs.com/`
[18] `http://www.simile-widgets.org/timeplot/`
[19] `https://square.github.io/cubism/`

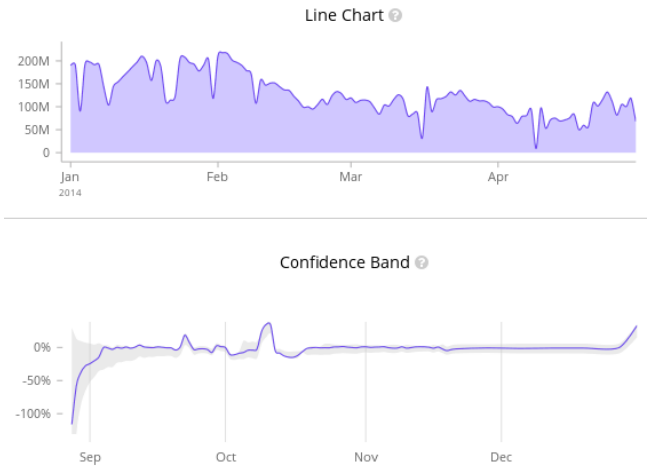


Figure 15: MetricsGraphics.js library

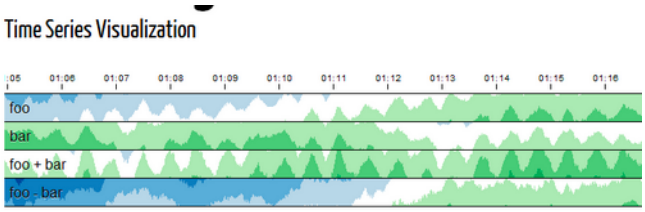


Figure 16: Time-series with Cubism.js library

4 Architecture solution

We investigated several architecture solutions, keeping in mind the requirements for this project :

- An easy to use interface (simple mechanics like drag and drop or button click) on Windows
- A reactive visualization
- A fluid visualization which can adapt to the quantity of data
- An easy to install application

4.1 Only data science language

Our first idea is to use only a language designed for data science and statistics (like R or Python). This has many advantages :

- It is paticularly adapted to Data Manipulation
- Huge data processing is relatively fast due to optimization
- There is visualization tools which are pretty accessible and easy to use

- There is a lot of resources for data manipulation and processing
- We could later include some more sophisticated computing on the data easily
- We only need to program in one language which improve maintainability of the system

However this solution seems limited for several reasons :

- It's a bit limited if we want to build a complete interface usable in a browser (but there are some libraries which could allow us to build web interfaces in Python)
- Available visualizations are limited by what the libraries can effectively display, which means if we need a very specific visualization it'll be harder to develop it (extending an existing library)
- The user needs to install Python or R on his computer in order for the scripts to run

4.2 Only web technologies

In the second solution we considered using only the Web technologies (HTML, CSS, JavaScript) for the application. This would have had some advantages :

- It's fairly easy to build complete interfaces with these technologies
- We can build a reactive interface and visualization
- D3.js would allow us to build pretty much any visualization without much trouble
- It's rather fluid to render charts using SVG, even if the data is large and/or changing
- There is a lot of resources for building interfaces and visualization from scratch or using external libraries
- The three main languages separate clearly the different aspects of the application (presentation of the data using HTML, styling using CSS and the logic and event handling in JavaScript).
- Everything can run inside the browser, the user doesn't need to install anything except an up-to-date browser

However we think this solution might not be the best for the following reasons :

- Data manipulation is not as straight forward as in languages designed for this task (there are some libraries to do it but JavaScript can not compete with R or Python in this domain)
- We may suffer from some slowness if we have computing tasks on huge datasets (again there are tools to overcome this : the use of web workers, optimized JavaScript like Emscripten, asm.js ...)

4.3 Hybrid approach

In the final solution we investigated, we thought about using both a language designed for data processing and manipulation in the back-end and a language (several ones) for data visualization in the front-end. We could then have the advantages of both the previous solutions :

- Fast and easy data manipulation and preprocessing
- Fast and reactive visualization
- Complete and simple interface

Obviously this would not come without some disadvantages :

- We need to spend some extra time in development in order to connect the two.
- The technology stack would be a bit more complex than in the previous solutions.
- The back-end language still needs to be installed. However, we could potentially make the back-end part a distant service, for multiple users. By doing so, the end user would only needs to use his browser to navigate to the address of the application in order to use it, without installing anything.

Criteria	**Python/R only**	**JavaScript only**	**Python/R and JavaScript**
Data Visualization	+	+++	+++
Preprocessing	+++	+	+++
Interactivity	++	+++	+++
Ease of Installation	++	+++	++

Table 1: Programming Language comparison

5 Technologies

5.1 Data Science : R or Python

5.1.1 R : presentation

R is a language and environment for statistical computing and graphics. It provides a wide variety of statistical techniques, such as linear and non linear modeling, classical statistical tests, time-series analysis, as well as graphical techniques.

R is an integrated suite of softwares facilities. It is mainly designed for manipulation, calculation and graphical display. For instance, it includes :

- an effective data handling and storage facility
- a suite of operators for calculations on arrays, as well as matrices
- a large, coherent, integrated collection of intermediate tools for data analysis
- graphical facilities for data analysis
- a well-developed, simple and effective programming language which includes conditionals, loops, user-defined recursive functions and input and output facilities.

The language syntax has a superficial similarity with C, but the semantics are of the Functional Programming Language variety.

Time Series analysis using R With its extensive facilities, R is a solid option for time series analysis. R can indeed convert numeric vectors to time series objects. These times series can then be plot, manipulated and cleaned to obtain understandable results. Among manipulations, we can decompose both seasonal and non-seasonal data depending on the expected model. There are also aditional tools for reading and writing data in different formats, including CSV.

5.1.2 Python : presentation

Python is a high-level programming language for general-purpose programming. It is an interpreted language and has a specific design for code readability, and a syntax that allows programmers to express concepts in fewer lines of code than other languages such as C++ or Java.

Time Series analysis using Python pandas is an open source library providing high-performance, easy-to-use data structure and data analysis tools for the Python programming language. Among the library highlights, we can enumerate the following ones : it has a fast and efficient DataFrame object for data manipulation. It has tools for reading and writing data between in-memory data structures and different formats, including CSV, as well as functions for flexible reshaping and pivoting of data sets Among the time series-functionalities we can enumerate date range generation and frequency conversion ...

5.1.3 R vs Python : our choice

See Table 2 on page 22.

5.2 Visualization : D3 only or Libraries

5.2.1 JavaScript and D3.js : presentation

JavaScript, often abbreviated as JS, is a high-level, dynamic, weakly typed, and interpreted programming language. Alongside HTML and CSS, it is one of the three core technologies of World Wide Web content production. JavaScript is used to make interactive web pages and provide online programs. JavaScript

Criteria	Python	R
Current Version	3.4.3 / 2.7.9	3.1.3
Last Update	February 2015 / December 2014	March 2015
Release year	1991	1995
Purpose	Emphasizes productivity and code readability. The closer you are to working in an engineering environment, the more you might prefer Python.	Focuses on better, user friendly data analysis, statistics and graphical models. The closer you are to statistics, research and data science, the more you might prefer R.
Community	More adoption from developers and programmers.	More adoption from researchers, data scientists, statisticians quants.
Usage	Used when data analysis tasks need to be integrated with web apps or if statistics code need to be incorporated into a production database.	Used when the data analysis tasks require standalone computing or analysis on individual servers.
Data handling capabilities	NumPy and pandas for data analysis	Huge number of packages, readily usable tests. Usable for basic data analysis without installing packages.
Pros	IPython Notebook, A General Purpose and Multi-Purpose language	Graphical Capabilities Ecosystem
Cons	Missing packages compared to R	Slow

Table 2: Python vs. R

combined with HTML and CSS is a great stack to create interactive interfaces which are OS-agnostic for the end-user. Also it doesn't require anything beside a browser to use the web-app.

On top of JavaScript there is D3.js, a library for manipulating documents based on data. It mainly uses SVG to display data in a fast and efficient way. It is the go-to library to build data-visualization on the web. Other library and tools for data-visualization are often built on top of D3, so the library is de-facto a standard for this kind of task.

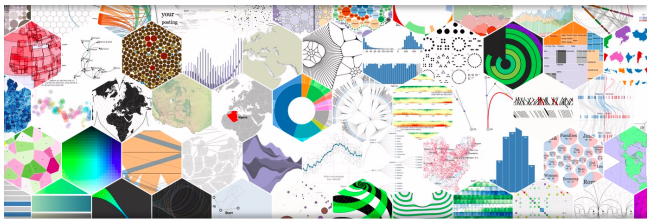


Figure 17: D3.js library

5.2.2 Libraries : presentation and examples

D3.js is a low level library, which means there are many other libraries built on top of it, some are general purpose libraries and other are more specific to a type of graph.

We think that since the graphics for the application are not necessarily standard, specific libraries are a better fit since it would be hard to find a general library that will be able to handle all of our cases (for example c3 is a general purpose library for data visualization built on top of D3 but it does not have a parallel coordinates type of graph).

However, there are many libraries dedicated to a type of graph in particular. For parallel coordinates for example, *parcoords*[20] provides an easy-to-use implementation of the chart with nice additional features like reordering of the columns, coloration, progressive rendering...

5.2.3 D3 only vs Libraries : our choice

We drew up a list of advantages and disadvantages of using plain D3 versus using specific libraries.

See Table 3 on page 25.

[20] `http://syntagmatic.github.io/parallel-coordinates/`

6 Our full solution

6.1 Presentation

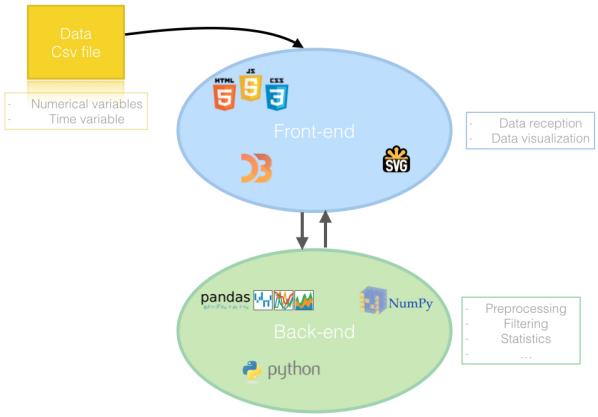


Figure 18: Presentation of our solution

6.2 Technical details

The solution that we think is best is using two parts for our project. We could have a back-end while the front-end would be using HTML, CSS, JavaScript and D3.

The back-end would implement a small Python server that will serve the webpage and expose a small API so that the front can communicate with the back.

To call the preprocessing functionalities the front-end could make request to specific URL or we could connect the front and the back using WebSockets to have easier discussions.

Dividing our application like this, would allow us to support more complex processing tasks and more complex functionalities in the future while being able to create an interface that suits exactly our needs.

This type of technology stack is used in several products and societies that offers data visualization tools. For example :

- Jobbing programmer and ex-research scientist Kyran Dale mentions how to clean data with pandas and display a data visualization [21]
- Toucan Toco, which offers data visualization tools, use python and pandas on their backend and JavaScript and D3 on their front-end.

[21] `http://kyrandale.com/static/talks/reveal.js/index_pydata2015.html`

	Plain D3	Specific Libraries	General Library
Pros	• D3 is very adaptative, every type of graph can be coded without much trouble. • We can connect the diffrent graphs very easily (filter one one and display into on another, ...) • We have a simpler technology stack. • We can follow the new versions of D3 without waiting for the libraries to be adapted. • There are many example for specific graphs in the D3 community. • We can fine-tune every graph easily. • We have a better understanding of the code since it's ours.	• Parallel coordinates and other graphs available out of the box. • Additional features already available (ex : coloration, adaptive rendering, ...). • Could potentially give back to the library by adding functionalities.	• Many type of graphs already available • Only need to use one library • Could potentially give back to the library by adding functionalities.
Cons	• We need to spend more time developing.	• Some features don't correspond to our use-case. • Many features which are a bit specific are missing (we need to code it in D3 and then integrate it back into the library : possible extra cost in development). • Specific libraries for each type of graph can lead to a complex technology-stack. • We're stuck with the version of D3 used by the library.	• It's impossible to find one general library that contains all the types of graph we need, some of which are a bit specific • Adding functionalities may be harder since it's a big library • There is a lot of graphs that we won't use, so we import a lot of code for just some functionalities

Table 3: D3 vs Library