The ultimate guide to **graph visualization**

Cambridge Intelligence

Contents

In a data-driven world, it's crucial to be able to interpret and communicate complex relationships. Graph visualization transforms connected data into intuitive, interactive visual representations that power investigations.

This ultimate guide dives into the fundamental concepts of graph visualization, its practical applications, and the tools you'll need for it.

Using illustrations created with our data visualization toolkits, we'll show you:

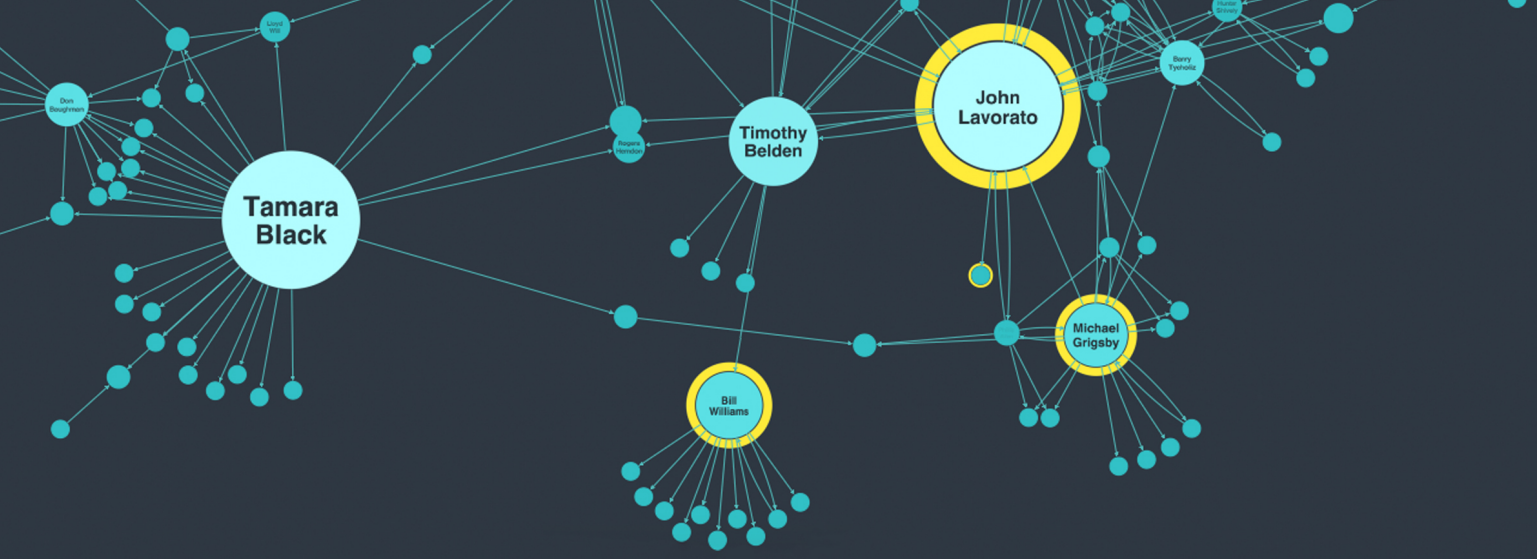
- How **artificial intelligence techniques** are revolutionizing graph analysis and visualization
- Practical **UX tips and strategies** to optimize usability and engagement
- Advanced **graph data visualization techniques** to uncover the most complex insights
- How **timeline and geospatial visualization** bring a whole new dimension to dynamic data

Who is this white paper for?

Product managers whose product UX relies heavily on connected data.

Tech managers and developers who want to create a seamless graph visualization experience.

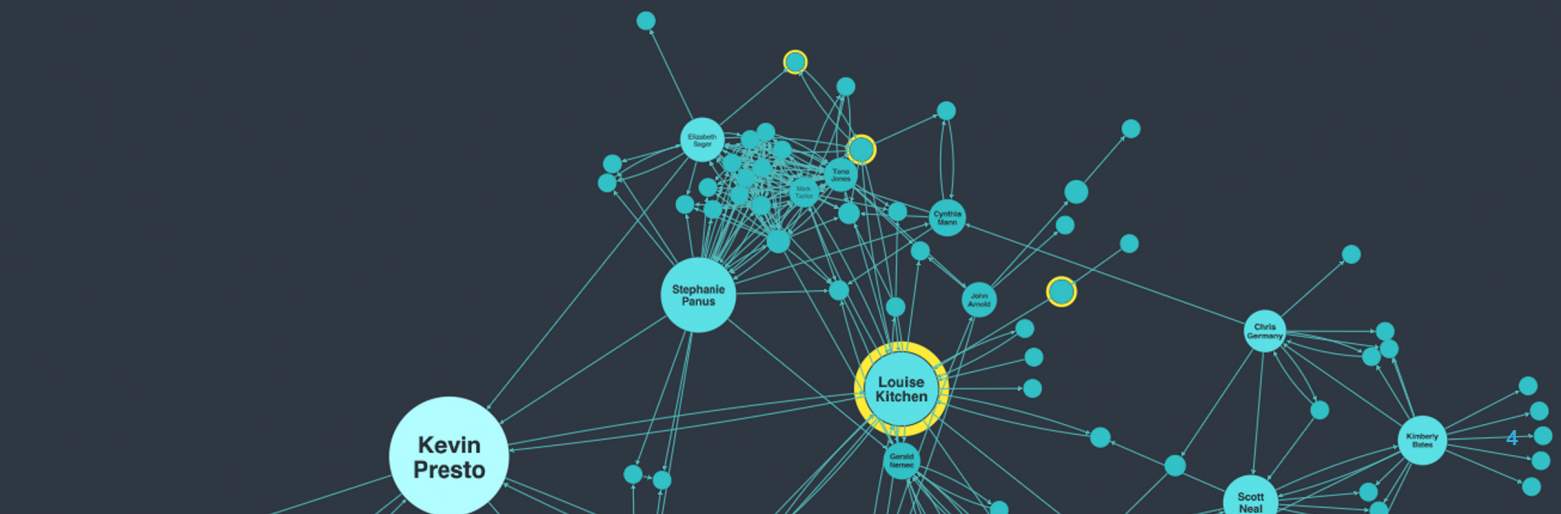
Executives who recognize the value of keeping their organizations ahead of the market with the latest visualization practices.



What is graph visualization?

Graph visualization, sometimes called ‘link analysis’ or ‘network visualization’, is the process of visually presenting connections (called links or edges) between entities (called nodes or vertices) and properties.

No matter how small the dataset is, if connections exist in the data, there’s value in visualizing them.



Why visualize graphs?

Graph visualization lets analysts and investigators identify trends, outliers and patterns of behavior, helping them make quick and smart decisions. It works because it's:

Intuitive

The node-link model mimics the way we naturally perceive and understand relationships between entities.

Fast

Our brains are quick to recognize trends, patterns and outliers when data is presented in a tangible format.

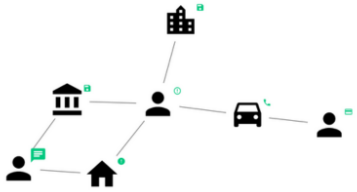
Flexible

The world is densely connected. If there's an interesting relationship in your data, you'll find value in graph visualization.

Insightful

When you explore an interactive graph visualization, you gain deeper knowledge, understand context, and can ask more questions.

Some popular real-world use cases

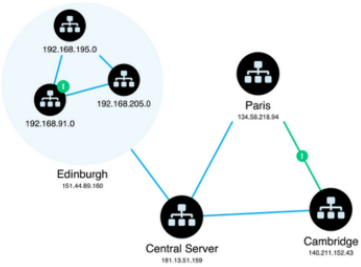


Security & intelligence

Distilling complex connected data into critical intelligence and insight

Anti-fraud

Detecting or investigating fraud in finance, insurance or online activity

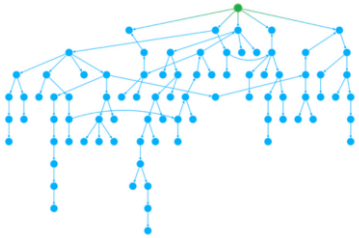


Cyber security

Tracking the behavior of cyber threats and analyzing incident forensics

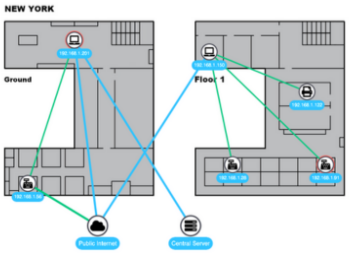
Law enforcement

Enabling detailed pattern of life and behavioral analysis



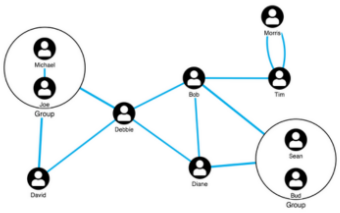
Compliance

Ensuring regulatory compliance through effective data analysis



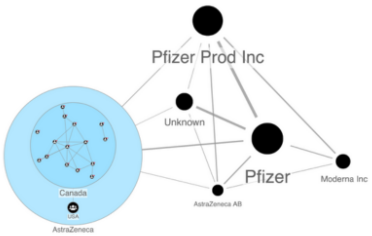
Infrastructure

Monitoring performance and faults, plus root cause analysis



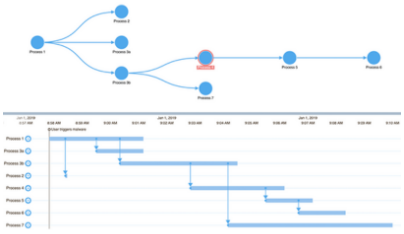
Customer 360

Understanding your customer behavior better



Pharmaceuticals

Analyzing connections between agents, diseases, drugs & trials



Supply chain

Uncover inefficiencies and manage regulatory compliance

Can AI do all the work for us?

Artificial intelligence is changing the way organizations think about data analysis.

AI can do a lot of the heavy lifting for analysts and investigators working with connected data – helping them detect, understand and even predict risks and threats.

But if AI can analyze information – why visualize data? And if AI can make recommendations, does that let humans off the hook?

The ideal approach is to apply machine learning to give structure to your data, and then let human managers analyze and explore it using an interactive visualization.

We call this the visualization/AI intelligence cycle, which has three stages:

- **Detect**

  AI software gathers, cleans and structures data from disparate sources and silos. It uses machine learning and pattern recognition to make recommendations and raise alerts.

- **Investigate**

  Interactive graph visualization presents these insights in a way that makes it easy for investigators to navigate and analyze, turning it into actionable intelligence.

- **Prevent**

  Investigators use what they've learned to inform the next set of queries and rules they feed into the system – perhaps using natural language prompts to speed up the process.

Download our guide to graph visualization & AI

Find out how artificial intelligence techniques are revolutionizing graph analysis and visualization.

Building the best application

Choose the right tech for the job

Open source code libraries

This option offers great flexibility. You can build a custom application and deploy it to anyone without additional costs. But if you have any technical problems, or bugs appear, they'll have to dig into the code to find the bugs themselves - or wait for the community (which may no longer exist) to implement fixes. Open source code libraries are often very basic or low level. That means they could lack the advanced functionality your users will need to explore and understand complex graph datasets at scale.

Off-the-shelf graph visualization applications

Off-the-shelf solutions allow users to navigate and query their graph data without any programming. They're intuitive tools for code-free investigation and insight, but they might not meet your specific functional requirements. Most applications are standalone - you can't embed them easily into your products and tools, so you'll disrupt your users' workflows.

Commercial graph visualization software development kits (SDKs)

Commercial SDKs like ours offer a solid compromise between an open source library and an application. They require some coding, but they come with detailed documentation and expert support for a faster developer experience. They'll also provide advanced functionality and scalability, making it quick and easy to build high-performance graph visualization tools.

Download our buyer's guide

This practical, impartial guide to choosing the right graph visualization technology for your project includes a handy comparison template.

Do you *need* a graph database?

You might gravitate towards graph databases for your graph visualization project – they have the word "graph" in them, after all.

However, they're not a prerequisite for visualizing graph data...

If you know you'll be running complicated queries through more than a couple of layers of data, then it's likely that a graph database will handle the complex relationships in your data most efficiently.

But you might prefer to stick with the tried and trusted technology you already rely on – and that could work out for you. Many of our customers have used non-graph data stores to build applications with all the speed, scale and insight they'd get from a graph database.

There are plenty of advantages to using a graph database...

They're a natural habitat for complex connected data: Many of our customers use relational databases, which aren't optimized for graph-related tasks. A graph database natively represents connected data and relationships, providing faster access and consistent response times.

You'll save time on mapping and modeling: Our brains work by making connections. Your design model will probably start with a node-link sketch. Working in a graph database means you can take that whiteboard model and apply it directly to your schema, with relatively few adaptations.

Schema flexibility: Graph databases allow your data model to evolve without disturbing the wider database structure. They're great at handling additions, gaps and anomalies in your data that a traditional tabular database might find tricky to accommodate.

They speak the local language: By far, the biggest advantage of a graph database is graph query languages. They're designed specifically for graph data, and let you frame your query in terms of the problem you're trying to solve, rather than creating an obstacle course of SQL queries.

But they could bring you some challenges...

Too many databases: If you're updating an existing system, it's likely that you're taking data from one or more legacy sources and copying it to a new graph database. Now you have yet another database to scale and maintain alongside your existing data stores.

Too many nodes: People using graph databases are prone to "overnodification" – the temptation to add every entity and property as a node. A graph database makes life easier at the modeling stage, but your visual model shouldn't always match the model of data in your database.

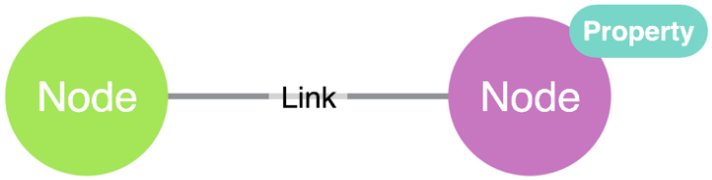
Create the perfect model

Data modeling is when you decide how to represent the entities, relationships and properties in your data. It's easy to skip over, but a well-designed data model is the foundation of your graph visualization experience. Taking time to get it right will help users understand their charts, and make your later design decisions much easier.

Graph data modeling

During the graph data modeling process, you decide which entities in your dataset should be nodes, which should be edges and which should be discarded.

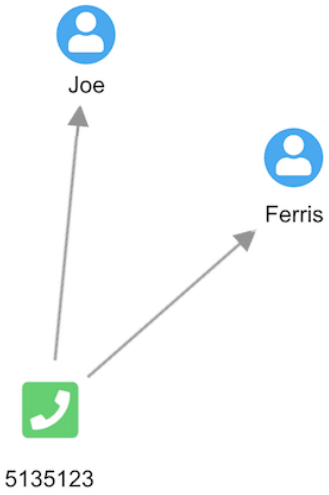
The result is a blueprint of your data's entities, relationships and properties. You can use that blueprint to create a visualization model for your charts.



Nodes are the fundamental units of our data. We design our entire model around these entities.

Links are the relationships between nodes.

Properties are descriptive characteristics of nodes and links, but aren't important enough to become nodes themselves. For example, a person's date of birth is an appropriate property.



Name	Phone number
Joe	5135123
Darryl	112657
Ferris	5135123
Stella	7838965

Designing a graph data model takes time, but it's worth getting it right, with a user-centered approach that your analysts will thank you for.

Graph data modeling best practice guides

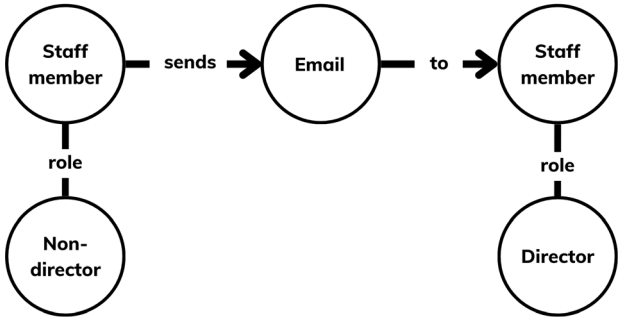
Visual data modeling

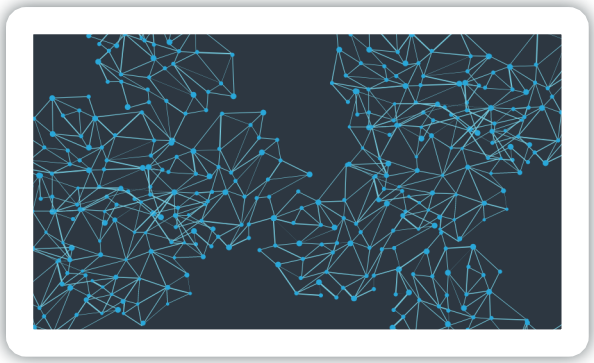
The visual model determines how the data model is represented on the chart. Design a model that's clear, clutter-free and helps users to instantly recognize what they're seeing.

The data model and the visual data model are rarely the same, and that's a good thing. Your data model is designed to work well for your database. Your visual model should be designed for your users, their data, and the questions they need to answer.

Don't add properties to your model just because they're in your database - you need to make decisions about what's going to add meaning to the visualization, and keep your chart clutter-free.

Let's say you're designing a social network analysis application. Your data model might include entities like staff members, job titles and emails. You could model this visually as:





To validate your graph data model, think about your users. What do they need to know? For example, the above model won't help a user that wants to know how many emails a staff member has sent, or to get an overview of the hierarchy in the organization.

By modeling the emails as directional links between staff members, and roles as properties with assigned node customizations, we can focus on what's important to our investigation:

Understand your users

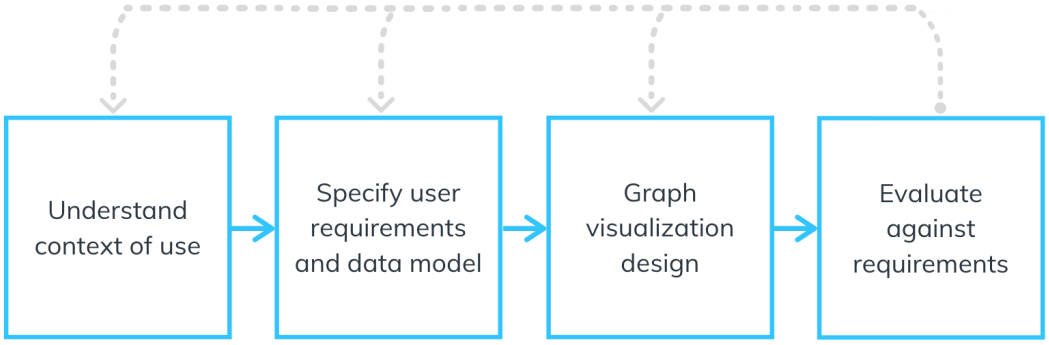
User experience (UX) design

For your graph visualization application to be a success, you need to do more than choose the right data store and front-end technology. You need to think about UX.

'UX design' is often used interchangeably with terms such as 'user interface design' and 'usability'. However, while usability and user interface (UI) design are important aspects of UX design, they are subsets of it – UX design covers a vast array of other areas, too.

Before you get started with any design project, you need to get the basics down. That means understanding your user. When you understand their problems and questions, you'll be able to design an effective visualization strategy.

UX design focuses on understanding and delivering what users want. The aim is to make them feel good about working with your graph visualization applications.



The design decisions you make will depend entirely on your own scenario but there are four cornerstones you should keep coming back to. With each decision, ask yourself if your UX is:

Intuitive – an intuitive experience is all about trust between an application and its users. Your users need to be able to rely on their graph visualization as an accurate representation of their data.

Consistent – users want consistency in the look and feel of your application. Don't forget the other applications in your users' stack, too - not just the visualization component.

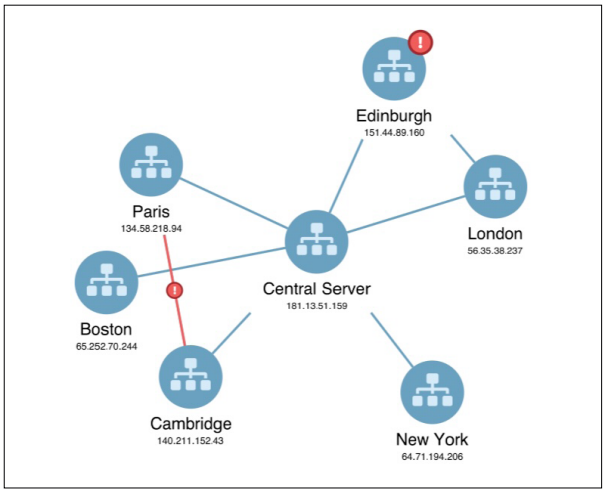
Traceable – to trust their graph visualization, the user needs to understand how it was generated. It might be tempting to run filtering, scoring and layouts entirely in the backend, but showing those processes will reassure your user that their results are accurate and trustworthy. Animation is a great way to do this.

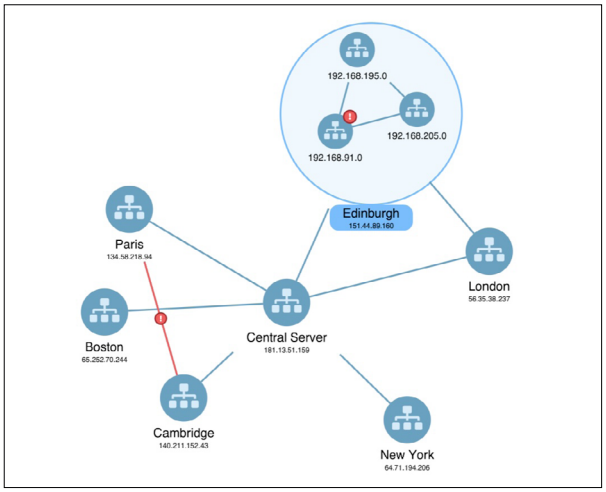
Reversible – don't leave your users in fear of an accidental click trashing an afternoon's work. Give them a quick and easy way to undo and redo their actions.

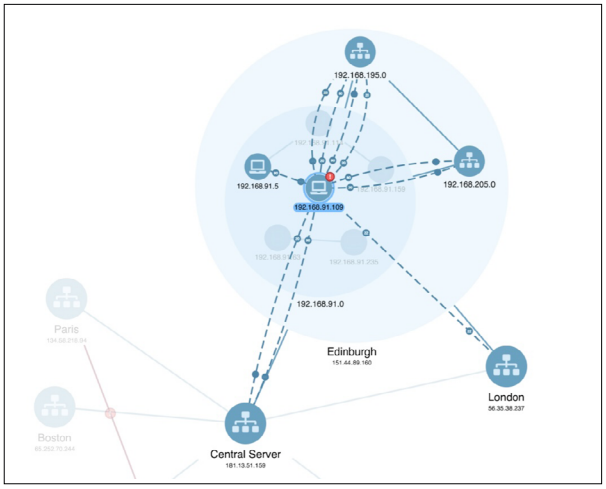
Chart interaction

Your users will have a far more insightful and enjoyable experience if they can manipulate and explore their visualization. You can decide how that interaction works, but remember the four cornerstones.

This fictional visualization of a company IT network displays offices at different locations and subnets within them, all speaking to a central server through which common services are accessed. Links represent communication between machines.







We've used combos to group network assets by subnet and location, so at the high level we get a really simple view of the topology.

Intuitive

It's easy for the user to understand the network, and pick out the two red alerts at a glance.

Consistent

The user expands the Edinburgh combo and starts to drill down to the source of the alert. The interactions along the way are so intuitive, the user knows what to do almost without thinking.

A double click opens a combo, and a single-click highlights nodes of interest and their key connections. The user can pan to a different area of the chart, or zoom in closer.

And at each level of detail, the red "alert" color coding signposts the user in the right direction as quickly as possible.

Traceable

Every time the user interacts with the chart, animations transform the chart smoothly in a way that's easy for the user to follow.

Reversible

The user can retrace their steps for a better understanding of the chart, or refresh the entire visualization to investigate a different question.

User interface (UI) design

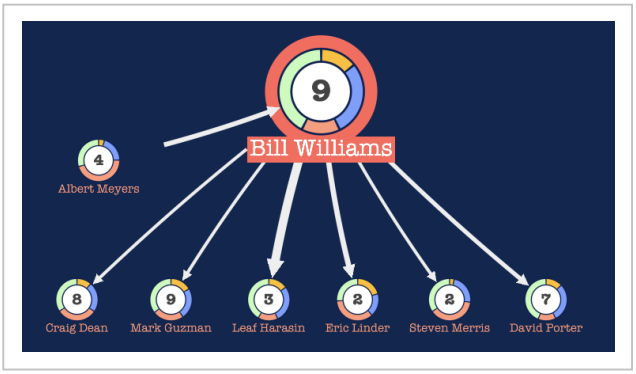
UI design is an essential component of UX. It focuses on the intuitive interactions and beautiful styling that make your application insightful and enjoyable for users.

With the right UI, your visualizations will:

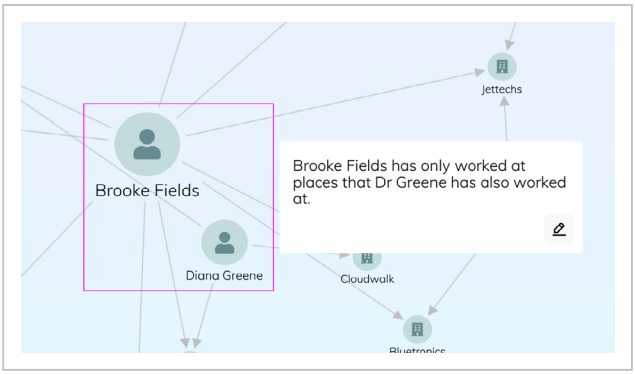
- Stand out from the crowd
- Show the right information at the right time
- Fit in with your product's design language
- Bring your UX designer's vision to life without compromise

Customized styling

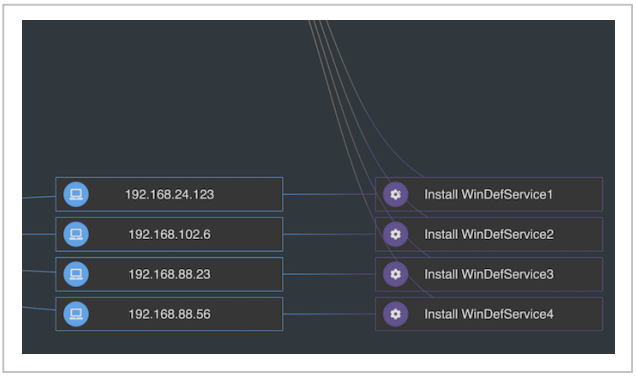
Visuals aren't just about aesthetics. Clever customization helps users follow the story of their data for better analysis.



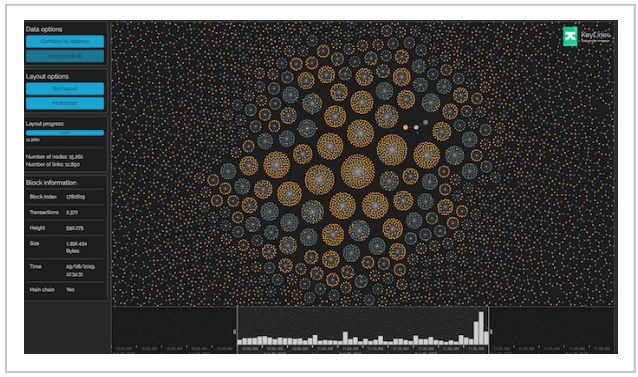
Formatting: Highlight key players by coloring or resizing nodes, or show numeric proportions of data, such as different centrality measures, with donuts around the nodes.



Glyphs, labels and annotations: Provide additional information, communicate interesting characteristics and help differentiate between chart items.



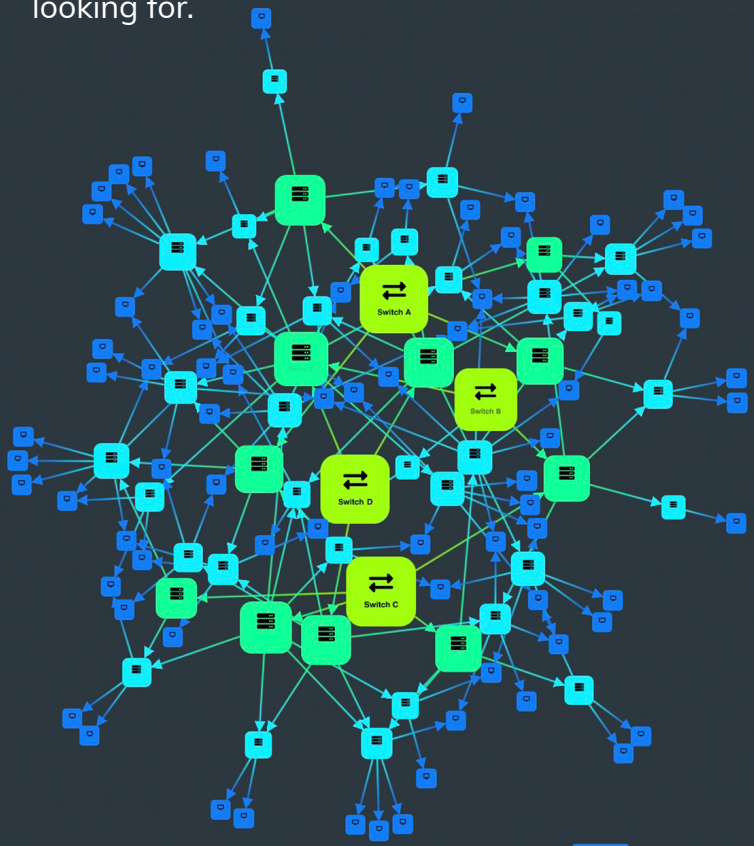
Fine-grain detail: Think carefully about every visual attribute, from arrow-head size, to label height, to link behavior when it approaches a node.



Charts: Customize the entire chart to meet the requirements of your application – including the background color, watermark text or image, chart logo and navigation controls.

Graph layouts

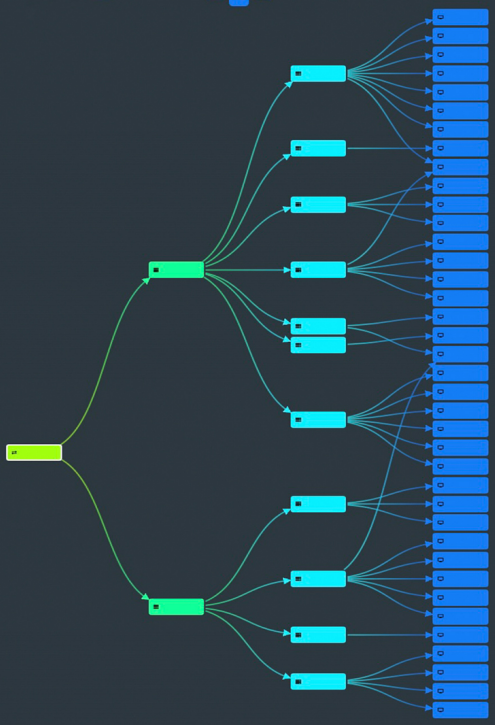
Graph layouts are sophisticated algorithms that decide where nodes and links are placed on a chart. A good layout goes beyond simply detangling links. It shows the patterns, anomalies, and clusters that direct the user towards the answers they're looking for.



These examples show our two most commonly used automated graph layouts.

The **organic layout** uses clever adaptive behavior to accommodate network changes through small adjustments to the chart, so users don't lose their mental map of the data.

It's our best-performing layout. The pattern is easy to understand, and helps reveal underlying structures.



The **sequential layout** is the most advanced way to explore tiered data.

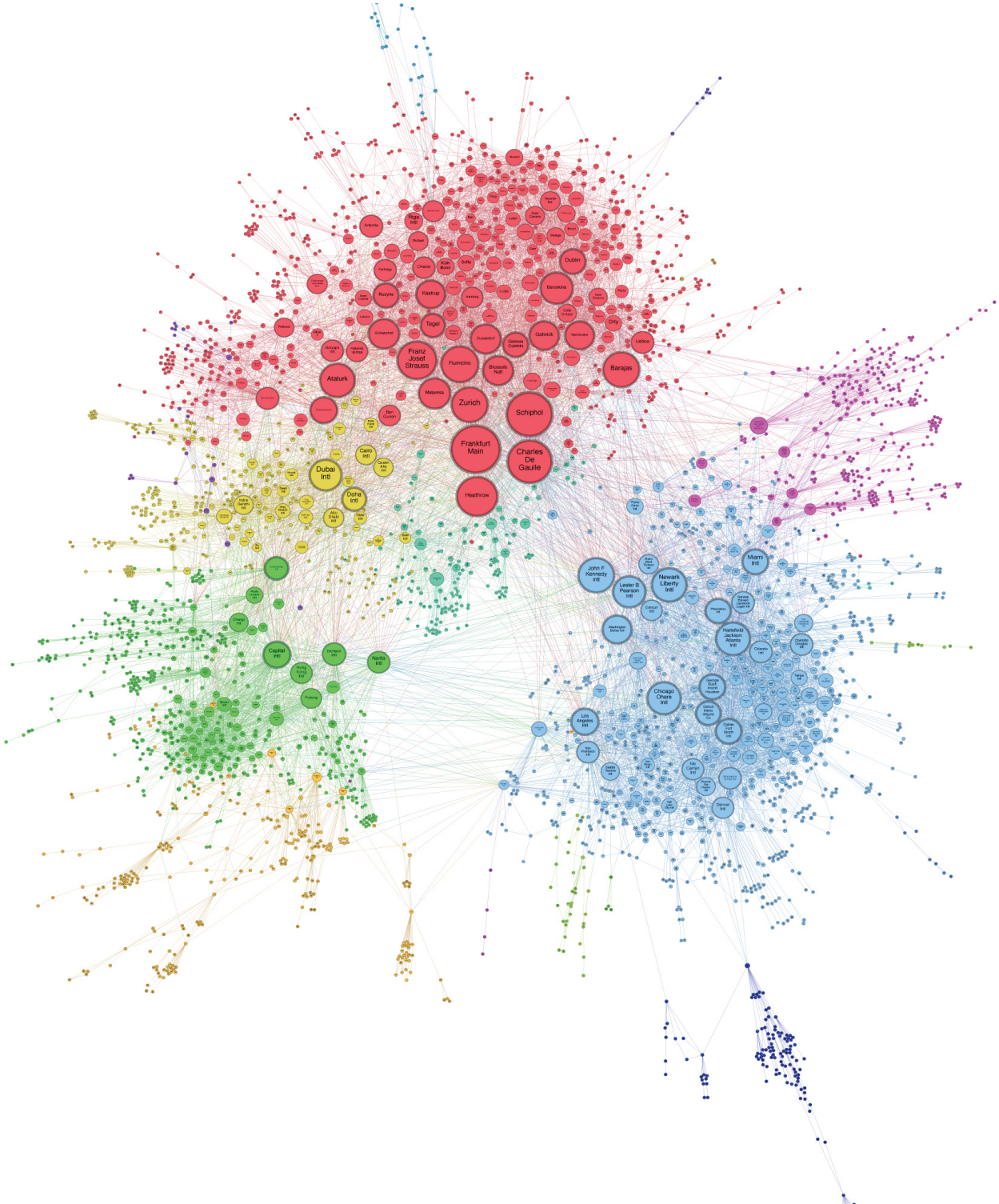
It's a popular way to display where there's a clear parent-child relationship between nodes, but where the nodes also have a set tier or level that needs to be communicated.

Explore more graph layouts

Adjust the signal-to-noise ratio

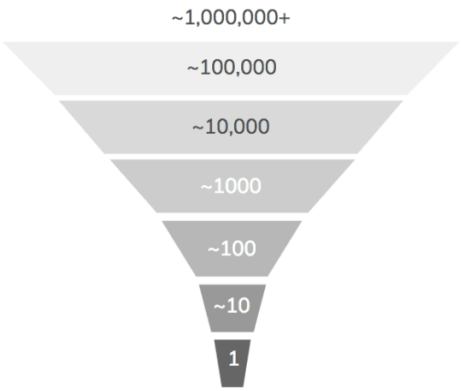
Your dataset may be complicated, but your graph doesn't need to look and feel complicated.

Let's look at how we can use the data funnel to amplify the elements of the chart that the user needs to see, and dial down the elements that aren't relevant to their investigation.



Put huge datasets through the data funnel

Using backend data management and front-end interactions, the data funnel condenses billions of data points into something a user can easily understand.



BACKEND FILTERING

BACKEND AGGREGATION

VISUAL DATA MODELING

FILTERING

COMBINING

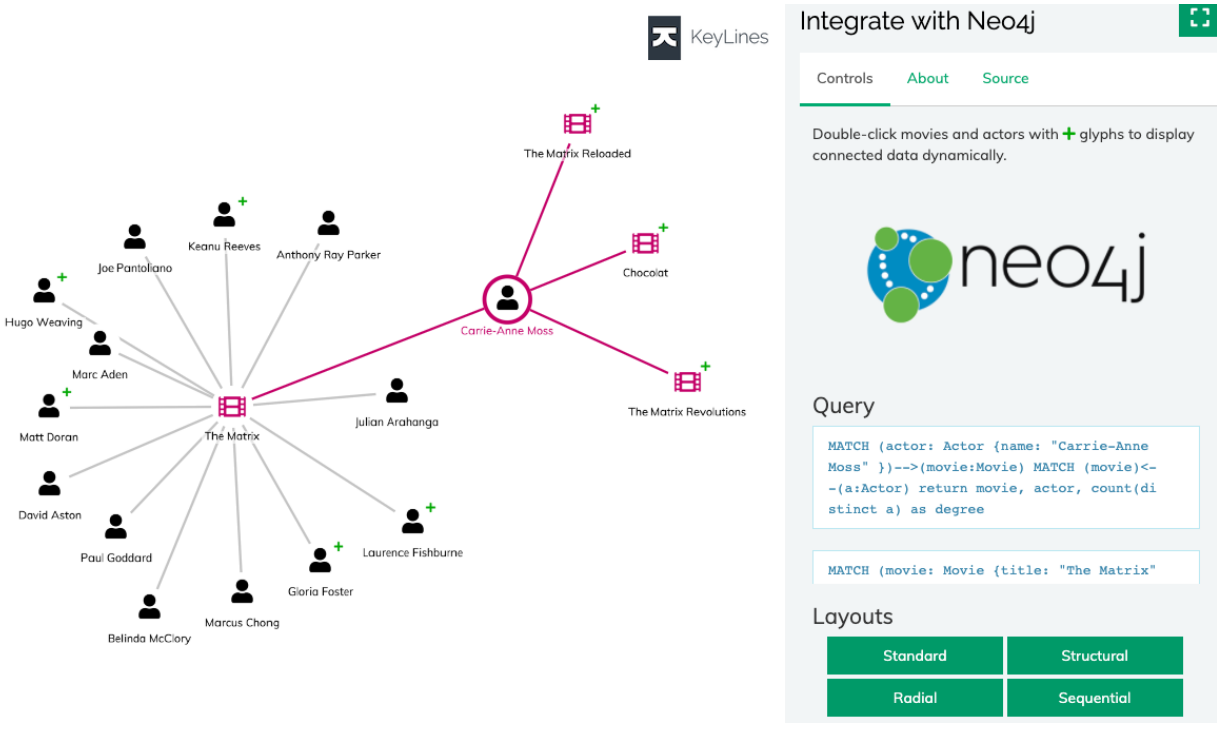
PRUNING

INSIGHT

1. Backend filtering

The first step of the data funnel happens on the backend. Chances are you've got a lot more data than you need to visualize. There's no point in visualizing your entire database - you'll run out of screen space and confuse your users. Instead, remove as much noise as possible, as early as possible.

You can bring filtering into your UI, too. Consider giving your users some visual ways to create custom filtering queries, like sliders, tick-boxes or selecting from a list of cases. In this example, we're using queries to power a 'search and expand' interaction:



But there's no guarantee that filtering through search will be enough to keep data points at a manageable level. Multiple searches might return excessive amounts of information that's hard to analyze. Filtering at the backend is effective, but it shouldn't be the only technique you use.

2. Backend aggregation

~100,000

Once filtering techniques are in place, you should consider aggregation. There are two ways to approach this:

1. Data cleansing to remove duplicates and errors: This can be time-consuming, but a handful of well-written queries could do a lot of the work for you.

2. Data modeling to remove unnecessary clutter.

With a few simple aggregation decisions, it's possible to reduce tens of thousands of nodes into a few thousand.

3. Visual data modelling

~10,000

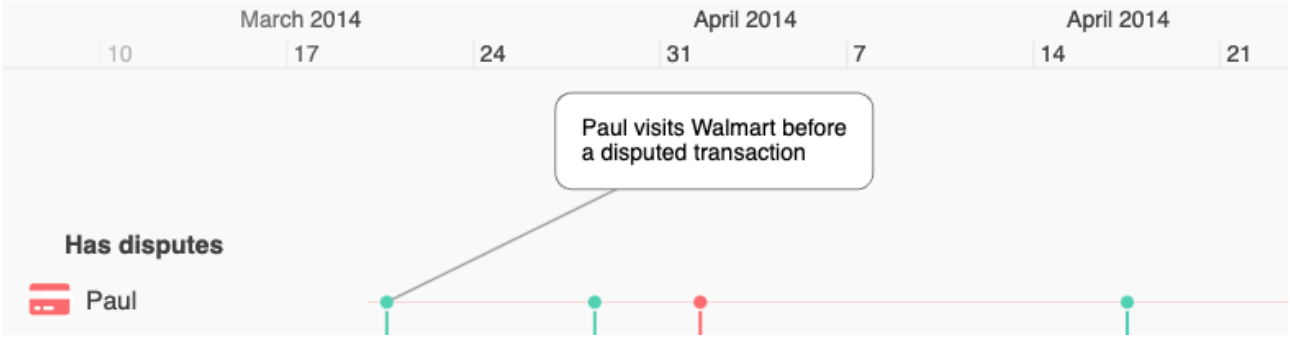
At this stage, you will have reduced 1,000,000+ nodes to tens of thousands.

This is where the power of visualization really shines. As we've seen earlier, we can use visual modeling to further simplify your data.

In this example we've used glyphs, link width and icons to represent the supply chain data stored in collapsed links:

glyphs to show the type of transport used, and the number of each type
link width to show the volume of activity between sites
icons to show the types of location

And here we've used **annotations** to add a new layer of meaning to the chart. Annotations can be user-driven or machine-driven: populated by the analyst during their investigation workflow, or automatically pulled from the data.



4. Filters and combos

~1000

Now we're down to a few thousand nodes, we can give users the tools to be a lot more selective themselves.

Once your users have the relevant nodes and links in their graph visualization, you should give them the tools to declutter and focus on the key items of interest.

Filters are great for this. For a great user experience and better performance, consider presenting users with an already-filtered view, with the option to bring in more data.

Another option is to use **combos** to group nodes and links, giving a clearer view of a large dataset without actually removing anything from the chart. It's an effective way to simplify complexity, but also to offer a 'detail on-demand' user experience that makes graph insight easier to find.

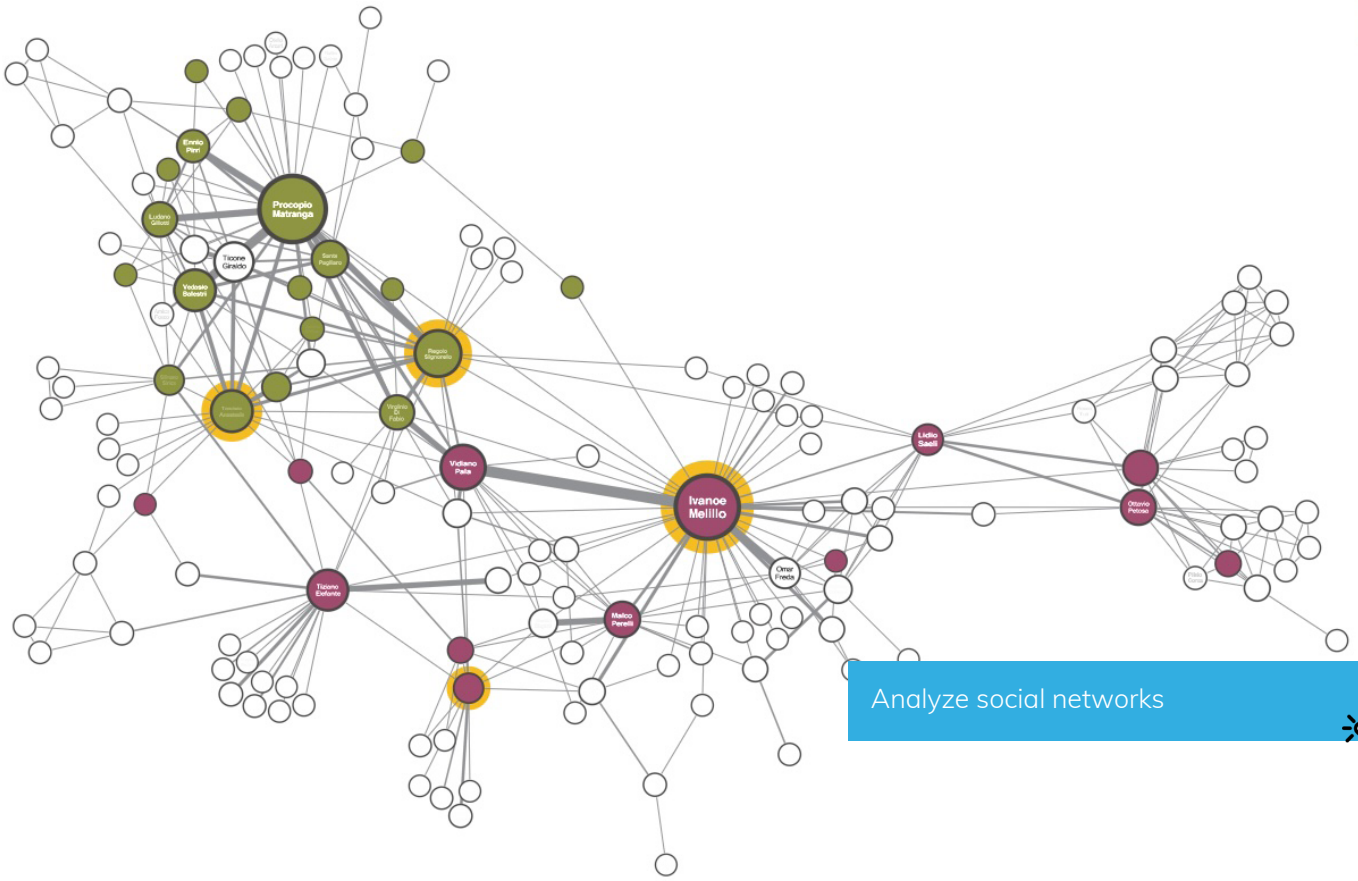
5. Pruning

~100

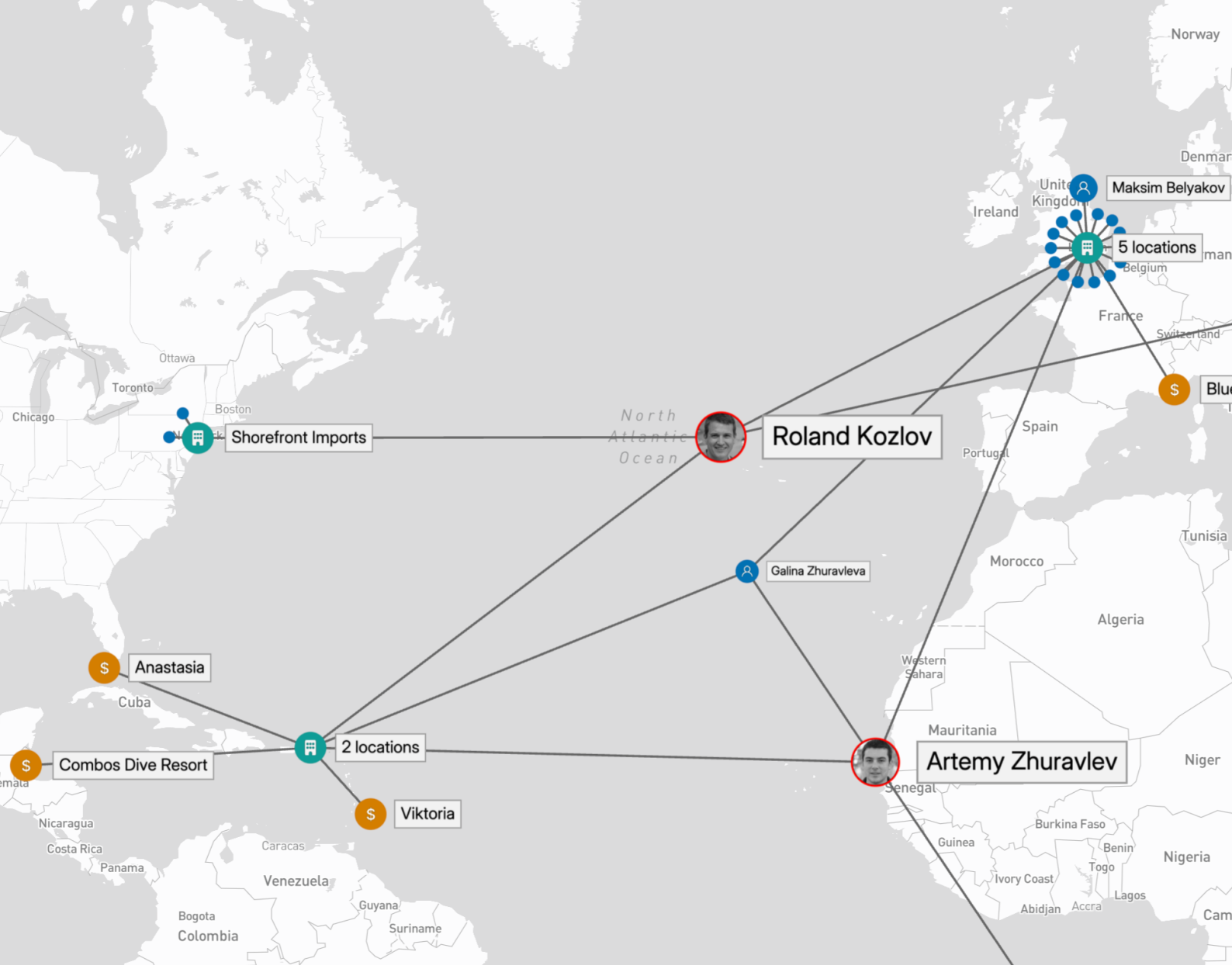
Social network analysis is a way to understand how networks behave, and uncover the most important nodes within them. Powerful social network visualization algorithms prune back through noisy social network data to reveal parts of the network that deserve most attention.

A node's **centrality** is a measure of its prominence or structural importance in a network. A high centrality score could indicate power, influence, control, or status. Finding out which is the most 'central' node can disseminate information in a network faster, stop epidemics, protect a network from breaking, identify suspected terrorists and much more.

Here's an example of how we can use a range of centrality measures to identify the most influential nodes in a social network:



Analyze social networks



Charting space and time

Geospatial data

Sometimes, data is easier to understand when you see it in the context of the world you live in.

Map-based data comes to life once you visualize it. When you combine geospatial charts with additional network or timeline data, you gain more insights from seeing entities and events in real-world scenarios rather than as columns and rows in a table.

If your data contains location information as latitude and longitude properties, it's worth trying out geospatial visualization. You can add ordinary images as scalable background maps, to give better context to visualizations and create powerful dashboards.

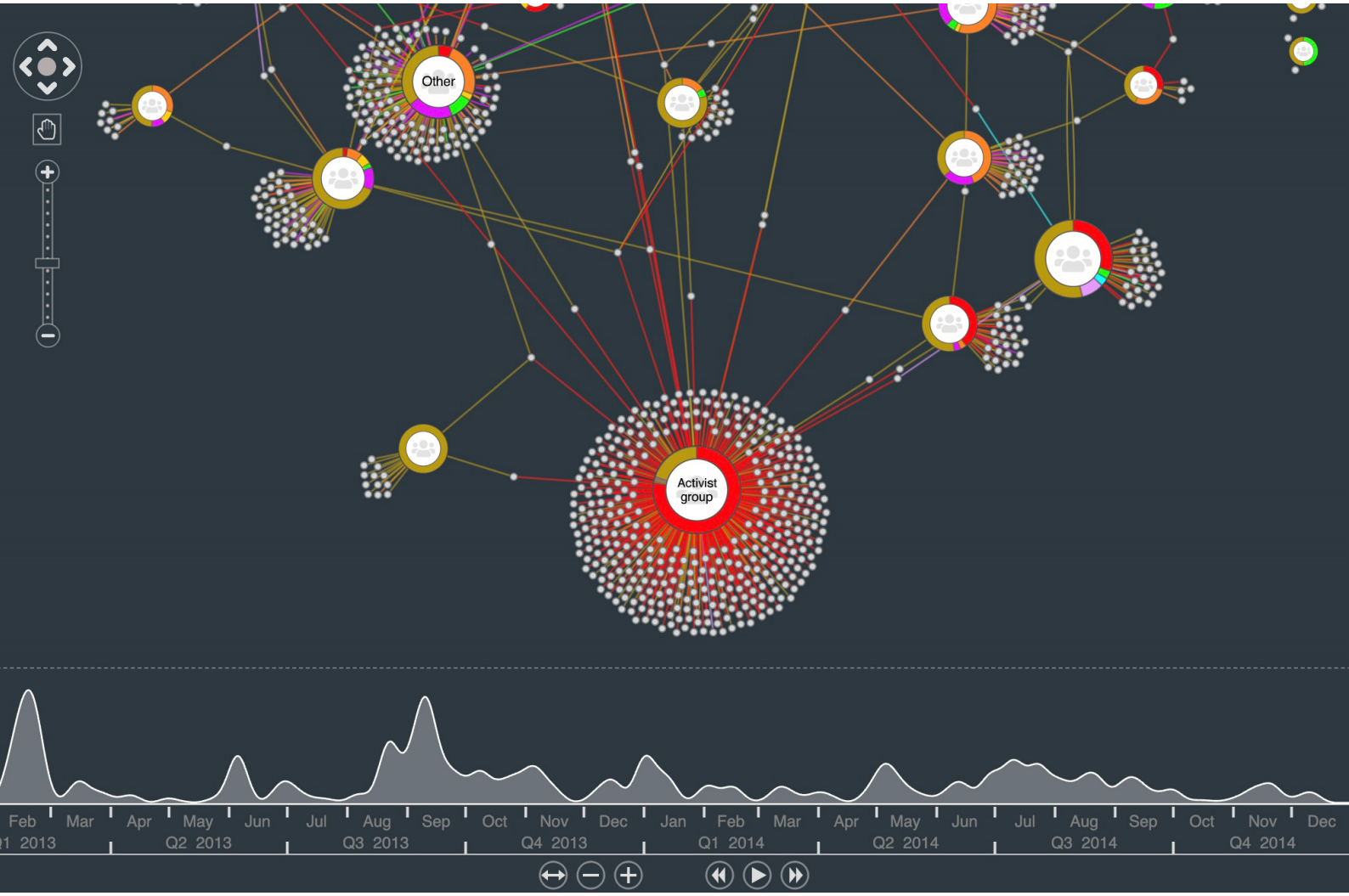
Discover geospatial visualization

Time-based analysis

Graph data is rarely static - networks evolve as connections are formed and broken. Understanding the time dimension in your massive connected data helps you uncover insight.

Exploring graphs with a time bar component helps analysts understand how networks form, cluster, fracture and dissolve over time. Users can explore their dynamic graph data in an interactive and intuitive way without getting overwhelmed.

The **time bar** in this example combines a histogram, showing overall graph activity, with trendlines, which show specific node or sub-network activity. Users can navigate, filter and play back their time-based graphs using scale and navigation controls.



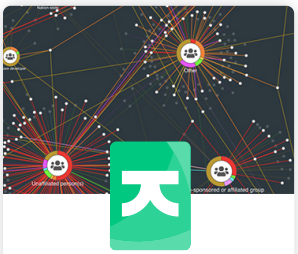
Request a free trial

Real-world data is complicated. Visualizing it shouldn't be.

Our developer toolkits make it easy to build powerful, customized user experiences for complex data.

cambridge-intelligence.com/try/

GRAPH



KeyLines

The graph visualization SDK for JavaScript developers

ReGraph

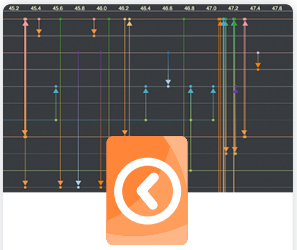
Hassle-free graph visualization for React developers

GEOSPATIAL

MapWeave

The geospatial visualization SDK that uncovers every connection

TIMELINE



KronoGraph

Advanced timeline visualizations that scale

Cambridge Intelligence

cambridge-intelligence.com USA +1 (775) 842-6665 UK +44 (0)1223 362 000
Cambridge Intelligence Ltd, 6-8 Hills Road, Cambridge, CB2 1JP