

Workflow Library: Titles and Summaries

1. Cluster and Calendar Based Visualization of Time Series Data

**Jarke J. van Wijk and Edward R. van Selow. *Proceedings of IEEE InfoVis '99*, pp. 4–9, 1999.
doi: 10.1109/INFVIS.1999.801851**

The workflow in this paper addresses the problem of understanding large univariate time series that contain patterns at several temporal scales at once, such as daily routines, weekday/weekend differences, holidays, and seasonal changes. The authors' central question is how an analyst can discover both normal and exceptional temporal patterns without having to specify in advance which calendar effects matter. Their answer is to reorganize the time series into comparable daily units, cluster days by similarity of their within-day profiles, and then display the resulting clusters in a calendar layout.

The workflow begins by splitting the original time series into day-long subsequences. Each day is treated as a pattern, such as a 24-hour electricity demand curve or a daily employee-presence curve. The analyst then chooses a distance measure that reflects the analytical question. A root-mean-square distance compares absolute profiles; a normalized distance emphasizes shape rather than magnitude; an offset-corrected distance removes slow trends; and a maximum-value distance highlights peak behavior. This choice determines what "similar days" means in the analysis.

Next, the daily patterns are clustered using bottom-up hierarchical clustering. Initially, every day is its own cluster. The algorithm repeatedly merges the two most similar clusters, producing a tree of increasingly broad groupings. Each cluster has an average daily profile, which summarizes the behavior of the days inside it. Rather than showing the full dendrogram, which becomes unreadable for a year of daily data, the workflow uses the clustering tree interactively. The analyst can view a chosen number of clusters, expand or contract a selected cluster, or recluster using a different distance measure or a selected time interval within the day.

The key visual step is to connect the clustering result to the calendar. Each day in a calendar view is colored according to its cluster membership, while the average profiles of the clusters are shown as line graphs. This makes the analytical logic explicit: the graphs show what kinds of daily behavior exist, and the calendar shows when those behaviors occur. Because the layout preserves months, weekdays, weekends, and holidays, the analyst can immediately interpret whether a cluster corresponds to Fridays, summer vacations, public holidays, seasonal demand, or unusual one-off events.

The workflow supports an iterative "overview first, then inspect" style. The analyst starts with a small number of clusters to identify dominant temporal regimes, then examines unusual days or subclusters. In the employee-presence example, the method reveals regular office-hour behavior, fewer people on Fridays and during summer, holiday effects, and a special early-departure pattern on December 5. In the electricity example, it reveals weekday/weekend structure, seasonal plateau levels, and a peak anomaly. The workflow

therefore relates directly to the paper’s research question: it exposes multi-scale temporal regularities and exceptions without requiring the analyst to predefine the relevant calendar categories.

2. Reducing Snapshots to Points: A Visual Analytics Approach to Dynamic Network Exploration

Stef van den Elzen, Danny Holten, Jorik Blaas, and Jarke J. van Wijk. IEEE Transactions on Visualization and Computer Graphics, 22(1):1–10, 2016. doi: 10.1109/TVCG.2015.2468078

This paper proposes a workflow for understanding how a network evolves over time. Its research question is how analysts can identify stable states, recurring states, outlier states, and transitions in dynamic networks without relying only on animation or small multiples. The core idea is to treat each temporal snapshot of a network as a single object, convert it into a high-dimensional point, project those points into two dimensions, and visually inspect the resulting geometry of network evolution.

The workflow starts with discretization. If the input is an activity log, such as timestamped contacts, messages, or transactions, the analyst divides time into windows and constructs one network snapshot per time window. Each snapshot contains the nodes and weighted edges active during that interval. The analyst can use overlapping windows so that adjacent snapshots change smoothly rather than abruptly. This step turns raw temporal events into a sequence of comparable network states.

The second step is vectorization and optional normalization. Each network snapshot is represented as an adjacency matrix, and the matrix is flattened into a row vector. The columns correspond to possible edges or derived structural attributes, and each row corresponds to one moment in time. Conceptually, every snapshot is now a point in a high-dimensional space where nearby points represent similar network topologies. Normalization, such as binarization, min-max scaling, or z-score normalization, can be applied depending on whether the analyst wants to emphasize edge presence, magnitude, variance, or outliers.

The third step reduces these high-dimensional points to two dimensions using methods such as PCA, MDS, or t-SNE. The projection is not interpreted as a conventional spatial map of nodes; instead, it is a map of network states. Clusters of projected points indicate time periods in which the network had similar structure. A compact cluster with consecutive times indicates a stable state, while a cluster containing points from different times indicates a recurring state. Isolated points suggest unusual network configurations, and trails between clusters reveal transitions.

The final step links this projection to detailed network inspection. One view shows the projected snapshots as points, often connected in temporal order and colored by time, hour, or another attribute. A second linked view shows the actual network for a selected snapshot using a node-link diagram or another network representation. The analyst can select clusters, inspect representative networks, brush nodes to see when they are active, and compare structural properties such as degree distributions.

The workflow therefore separates two analytical levels. The projection view supports reasoning about the evolution of the whole network, while the network view supports interpretation of individual states. In artificial examples, the method recovers known stable and recurring states. In high-school contact data, it reveals night states, daily loops, class-break rhythms, dense morning contacts, and unusual patterns such as exam periods. The workflow answers the paper’s research question by making dynamic network evolution visible as trajectories among states rather than as a sequence of hard-to-compare network drawings.

3. MobilityGraphs: Visual Analysis of Mass Mobility Dynamics via Spatio-Temporal Graphs and Clustering

Tatiana von Landesberger, Felix Brodkorb, Philipp Roskosch, Natalia Andrienko, Gennady Andrienko, and Andreas Kerren. IEEE Transactions on Visualization and Computer Graphics, 22(1):11–20, 2016. doi: 10.1109/TVCG.2015.2468111

The MobilityGraphs workflow is designed to analyze collective movement flows over space and time, such as movements inferred from geolocated tweets or mobile phone records. The paper’s research question is how analysts can understand typical spatial flow situations and their temporal variation when ordinary flow maps are too cluttered and when long time series make animation or small multiples ineffective. The proposed answer is to simplify the data spatially and temporally, then explore the simplified movement situations through coordinated visual views.

The workflow begins by representing mobility data as a time-varying directed weighted graph. Places are graph nodes, movements between places are directed edges, and edge weights represent flow magnitudes during each time step. Each time step therefore has a “spatial situation,” meaning a complete distribution of movements across the territory. Directly visualizing these situations produces severe clutter because many flows cross each other.

The first major analytical operation is spatial simplification. The authors build a “supergraph” summarizing flows across all time steps and apply an extended DBScan-like clustering algorithm. This algorithm groups places that are geographically close and strongly connected by flows. Unlike purely spatial clustering, it prevents nearby but weakly connected places from being merged. Unlike clustering each time step independently, it produces consistent spatial aggregates across time, which is essential for comparison. Small or weakly connected aggregates can be treated as noise. The result is a smaller set of meaningful regions and aggregate flows between them.

The second operation is temporal simplification. After spatial aggregation, each time step is described by the magnitudes of flows between the aggregate regions. These simplified spatial situations are then clustered, for example with k-means, so that time steps with similar movement patterns fall into the same temporal cluster. This reduces a long sequence of hourly flow maps to a manageable number of representative movement situations. Spatial simplification is performed first because clustering the original flow vectors would be too high-dimensional and unstable.

The visual exploration stage connects these results. A calendar view shows when temporal clusters occur, revealing daily and weekly rhythms. Small map thumbnails show representative aggregate flow graphs for each temporal cluster, with node size indicating presence and edge width indicating flow magnitude. A multidimensional-scaling layout can position cluster thumbnails by similarity. A difference view compares two selected situations by showing which regions and flows increase or decrease.

The analytical logic is iterative. The analyst adjusts clustering parameters, observes how the spatial aggregates and temporal clusters change, and uses visual comparison to interpret mobility routines. In the London Twitter case, the workflow reveals a monocentric pattern with morning flows toward the center, evening flows outward, quieter daytime and nighttime periods, and weekend differences. In the Abidjan mobile-phone case, it reveals a more polycentric structure and distinguishes morning, afternoon, evening, and anomalous periods. The workflow thus answers the paper’s research question by reducing clutter and temporal overload while preserving enough spatial and temporal structure to interpret mass mobility dynamics.

4. Temporal Event Sequence Simplification

Megan Monroe, Rongjian Lan, Hanseung Lee, Catherine Plaisant, and Ben Shneiderman.
IEEE Transactions on Visualization and Computer Graphics, 19:2227–2236, 2013. doi:
10.1109/TVCG.2013.200

This paper describes a workflow for making large temporal event sequence datasets analytically usable by progressively simplifying them. Its motivating problem is that electronic health record data often contain long, heterogeneous, noisy patient histories, while researchers usually need to answer focused questions about a subset of events and records. The workflow is built around EventFlow, which aggregates records with identical event sequences into a population-level visualization. When records are too varied, the display becomes visually complex, so the analytical task becomes one of reducing irrelevant or misleading complexity while preserving the logic needed for hypothesis generation and testing.

The workflow begins with an initial aggregated display of the raw event sequence data, often aligned by a sentinel event such as a prescription. This display reveals whether the data can be meaningfully summarized. If the visualization becomes “confetti,” with too many small visual elements, the analyst begins simplification. The paper introduces two complexity measures: the number of visual elements and their average height in the aggregate display. These measures help analysts judge whether simplification is improving the population overview.

The first simplification stage uses filters. Record filters keep only patients matching a query or remove irrelevant subgroups. Category filters hide event types unrelated to the research question. Time filters restrict the view to a meaningful window around the alignment point. Attribute filters remove events or records based on metadata such as dosage or patient characteristics. These operations help the analyst isolate the population and time span relevant to a hypothesis.

The second stage transforms the representation of events. Interval merging combines adjacent or slightly overlapping intervals, such as repeated medication prescriptions, into a single exposure period when that better reflects the clinical concept of interest. Category merging combines multiple event types into a broader clinical class. Marker insertion creates a special event at a meaningful point, such as the “new” LABA prescription, so that the analyst can align and reason around that occurrence without displaying every similar event.

The third stage generalizes these operations into a temporal Find-and-Replace workflow. The analyst specifies a temporal pattern using EventFlow’s query language, then replaces it with a more meaningful representation, inserts a marker, or deletes it. This makes simplification flexible enough to express study-specific logic. For example, concurrent LABA and inhaled corticosteroid prescriptions can be replaced by a single combined-treatment interval.

The workflow is iterative and interpretive. Analysts simplify, inspect the resulting aggregate, refine the simplifications, and, if necessary, reinsert detail. In the LABA case, the simplified display revealed that many patients had not received recommended prior treatments before a LABA prescription. In the opioid case, raw prescriptions were transformed into acute, intermediate, and chronic episodes, enabling researchers to test whether apparent acute users were misclassified chronic users. The workflow therefore supports the paper’s research question by turning overwhelming event data into analytically meaningful representations that expose population-level temporal patterns.

5. EventThread: Visual Summarization and Stage Analysis of Event Sequence Data

Shunan Guo, Ke Xu, Rongwen Zhao, David Gotz, Hongyuan Zha, and Nan Cao. IEEE Transactions on Visualization and Computer Graphics, 24(1):56–65, 2018. doi: 10.1109/TVCG.2017.2745320

EventThread presents a workflow for discovering and interpreting latent evolution patterns in collections of event sequences. The paper’s research question is how analysts can summarize many sequences that are similar in broad development but not identical in exact event order, and how they can identify stages in these developments while still retaining access to the underlying events. The workflow combines preprocessing, tensor-based analysis, interactive clustering, and coordinated visualization.

The workflow begins with raw event sequences associated with entities, such as patients, cars, or academic careers. Because real event data can be noisy and heterogeneous, the first preprocessing step filters events using a TF-IDF-inspired measure. Events that help distinguish one sequence from others are retained, while low-information events can be removed. Next, sequences are aligned to a common origin. The origin may be a first hospitalization, a first maintenance event, or another meaningful starting point. This alignment makes sequences comparable in terms of progression rather than calendar date.

The aligned sequences are then segmented into stages. Stages may be fixed time intervals, such as years, or event-count intervals, such as rounds of car maintenance. Within each

stage, events are folded into a summary representation. The result is encoded as a three-way tensor whose dimensions represent events, stages, and entities. This tensor records which events occur for which entities at which stages.

The analytical core is tensor decomposition. The system decomposes the tensor into latent “threads,” where each thread summarizes a characteristic evolution pattern across stages. Each thread has relationships to events, stages, and entities: it indicates which events are likely, when the pattern appears, and which entities are associated with it. This lets the method go beyond exact sequence matching. It can identify clusters of sequences that share latent structure even if their low-level event details differ.

The visualization stage turns these threads into an interpretable map of progression. Threads are drawn as horizontal or branching lines divided into stages. At each stage, events with high likelihood are shown as colored nodes on the thread. Similar thread segments can be interactively grouped into latent stage categories. A layout algorithm positions similar stage segments near each other while reducing line crossings, so analysts can see threads merge, diverge, or persist over time. The analyst can adjust the clustering level to explore coarser or finer stage categories.

Several linked context views support interpretation. An event flow view shows aggregated raw event sequences for selected threads. Entity and event lists reveal the underlying records and dominant events. A thread overview summarizes event-type composition. Brushing, highlighting, tooltips, and filtering allow the analyst to move between latent patterns and raw evidence.

The workflow is demonstrated on COPD patient records, car maintenance histories, and academic behavior records. In the COPD case, it identifies a branch of patients with increased later hospital admissions. In the car case, it relates oil type and diagnostic behavior to later repair patterns. The workflow thus answers the research question by combining high-level latent stage discovery with low-level event context needed for semantic interpretation.

6. EventAction: Visual Analytics for Temporal Event Sequence Recommendation

Fan Du, Catherine Plaisant, Neil Spring, and Ben Shneiderman. IEEE Conference on Visual Analytics Science and Technology (VAST), pp. 61–70, 2016. doi: 10.1109/VAST.2016.7883512

EventAction describes a workflow for prescriptive analysis of temporal event sequences. Rather than only describing what happened or predicting what may happen, the system helps analysts recommend what actions to take and when. The paper’s research question is what combination of algorithmic analysis and interactive visualization can help users review recommended actions and improve outcomes. The workflow is developed through a student-advising scenario, but the logic is intended for domains such as healthcare, e-commerce, and coaching.

The workflow begins with a current record, such as a graduate student’s academic history so far. The record is shown as a timeline organized by event categories and time periods.

In the advising case, the time axis is divided into semesters, and events include courses, publications, assistantships, milestones, and degree progress. This first view lets the advisor and student understand the current trajectory before comparing it to historical cases.

The second step finds similar archived records. EventAction compares the current record with historical records over the same initial time window. In the prototype, similarity is based on feature vectors counting events in each category, though the paper notes that richer similarity models could be used. The similarity distribution is shown visually, and the user selects a cohort of comparable archived records. This keeps the human analyst involved in deciding how broad or narrow the reference group should be.

The third step estimates potential outcomes. The system summarizes the outcome distribution of the selected similar records and compares it with the distribution for all archived records. In the student case, outcomes are job types such as software engineer, industrial postdoc, academic postdoc, or assistant professor. The user can choose a desired outcome, and the interface highlights information relevant to that goal.

The fourth step helps the user understand which actions may matter. A correlation view shows how event categories relate to outcomes among the similar records. For example, publications or research assistantships may correlate with becoming an assistant professor. An activity summary view then shows when successful similar records tended to perform those actions. The display can show all similar records, only those that achieved the desired outcome, or the differences that distinguish successful records from the broader cohort.

The final step is action planning. The user interactively adds planned future events to the current record, such as taking advanced courses, publishing papers, or serving as a teaching assistant. EventAction periodically recomputes the similarity-based outcome estimate and shows how the plan changes the probability of each outcome. The analyst can therefore iteratively refine the plan while combining domain judgment with data-derived patterns.

The workflow deliberately avoids producing a single automatic prescription. Instead, it presents evidence from similar histories, shows outcome associations, and lets the user construct a personalized plan. This directly serves the paper's research question: it turns temporal event sequence analytics into an interactive recommendation process where users can understand, question, and adapt the suggested actions.

7. From Movement Tracks through Events to Places: Extracting and Characterizing Significant Places from Mobility Data

Gennady Andrienko, Natalia Andrienko, Christophe Hurter, Salvatore Rinzivillo, and Stefan Wrobel. IEEE VAST, pp. 161–170, 2011. doi: 10.1109/VAST.2011.6102454

This paper presents a workflow for deriving meaningful places from raw movement trajectories. Its research question is how analysts can identify places that matter because certain movement events occur there repeatedly, when those places are not known in advance and may have arbitrary shapes, sizes, or overlaps. The workflow moves from trajectories to movement events, from events to significant places, and from places to aggregate movement patterns.

The workflow begins with movement tracks: sequences of time-stamped positions for moving objects. These raw tracks are semantically poor, so the first analytical task is to compute dynamic movement attributes. These can include speed, direction, acceleration, turn angle, distance to contextual objects, time of day, or relations to spatial and temporal context. The analyst then uses visual queries to extract movement events, or m-events, defined by conditions on these attributes. For example, low-speed trajectory segments may indicate traffic congestion; low altitude near the end of an aircraft trajectory may indicate landing.

The second step identifies significant places by clustering the extracted events. The paper uses density-based clustering because the goal is to find repeated concentrations of events while treating isolated events as noise. A custom distance function can combine spatial distance, temporal distance, and thematic attributes such as movement direction. This is important because events close in space may not be analytically equivalent; for example, congestion in opposite directions on the same road should be separated. In some cases the workflow uses two-stage clustering: first spatio-temporal clustering to find event episodes, then spatial clustering to unite repeated episodes occurring in the same location. Spatial buffers or convex hulls are then built around clusters to delineate significant places.

The third step aggregates events or trajectories using the extracted places. Events can be counted by place and time interval, producing time series that describe when events occur at each place. Trajectories can also be aggregated by places or pairs of places, producing statistics about visits, durations, flows, and movements between significant areas. This step turns raw tracks into structured summaries that can be analyzed at a higher level.

The fourth step analyzes these aggregates visually. Time graphs, maps, space-time cubes, temporal diagrams, and flow maps reveal when places are active, how intensity varies, and how places are connected. In the Milan traffic example, low-speed events are extracted, clustered by space, time, and direction, converted into congestion areas, and then analyzed to reveal morning and afternoon congestion patterns. In the French air-traffic example, landing and takeoff events are extracted, clustered to identify airport catchment areas and directions, then used to analyze airport activity and flows between airports.

The workflow directly supports the paper's research question by making places emerge from repeated movement behavior rather than from predefined administrative regions. It provides an ordered analytical logic: define relevant event types, find their spatial recurrence, aggregate movement around the resulting places, and then interpret spatio-temporal patterns.

8. Visually-Driven Analysis of Movement Data by Progressive Clustering

Salvatore Rinzivillo, Dino Pedreschi, Mirco Nanni, Fosca Giannotti, Natalia Andrienko, and Gennady Andrienko. *Information Visualization*, 7:225–239, 2008. doi: 10.1057/PALGRAVE.IVS.9500183

This paper describes a workflow for exploring large trajectory datasets through progressive clustering. Its research question is how analysts can use clustering to make sense of many

complex movement trajectories when no single similarity measure can capture all relevant trajectory properties. The proposed workflow applies simple, interpretable distance functions step by step, allowing the analyst to refine earlier results and build understanding gradually.

The workflow starts from the recognition that trajectories have multiple properties: origins, destinations, routes, timing, and movement dynamics. A universal distance function would be computationally expensive and difficult to interpret. Instead, the analyst chooses a distance function that matches a specific question. For example, “common destination” groups trips ending near the same place; “common source and destination” groups trips with similar starts and ends; and “route similarity” groups trajectories following similar paths. More complex functions can account for movement dynamics, but they are best applied only after the dataset has been reduced.

The clustering method emphasized in the workflow is OPTICS, a density-based algorithm that orders objects by reachability distance and reveals cluster structure without forcing every trajectory into a cluster. This is useful for movement data because many trajectories may be unique or noisy. The analyst inspects the clustering result visually, typically on a map, and interprets the clusters in relation to the spatial context. Dense coherent groups become meaningful movement patterns, while unmatched trajectories can remain noise.

The key feature is progression. The analyst first applies a cheap, interpretable clustering operation to the full dataset. In the Milan car-trajectory example, the workflow begins by clustering trips according to destinations. Because density varies spatially, the analyst adjusts parameters and performs clustering in stages: large dense destination clusters are identified and removed, then the remaining data are reclustered with lower density thresholds to reveal smaller destination regions. This handles uneven data density while maintaining interpretability.

After broad clusters are understood, the analyst selects subsets for deeper analysis. For example, trajectories ending in Milan’s city center are reclustered by route similarity. This reveals that many are short local trips, while relatively few longer trajectories share common routes. The analyst then uses this insight to formulate a new question about whether short local trips are common across the whole dataset. A cheaper source-and-destination clustering confirms that many trips begin and end near each other. Finally, the analyst excludes these local or round trips and applies route-similarity clustering to the remaining longer trips, identifying peripheral belt routes and radial inward or outward routes.

Visualization is essential throughout the workflow. Clusters are shown over maps, and aggregation techniques can summarize movement through significant places and aggregate moves. The analyst interprets each clustering result, decides which clusters to retain, exclude, or refine, and chooses the next distance function accordingly.

The workflow answers the paper’s research question by treating clustering not as a one-shot automatic result but as an interactive sensemaking process. Each clustering step has a clear semantic meaning, and the sequence of steps lets the analyst move from broad structure to detailed movement patterns while controlling computational cost and interpretive complexity.

9. Human-Steered Topic Modelling

J. Choo, C. Lee, C. K. Reddy and H. Park, “UTOPIAN: User-Driven Topic Modeling Based on Interactive Nonnegative Matrix Factorization,” *IEEE Transactions on Visualization and Computer Graphics*, 19(12), pp. 1992–2001, Dec. 2013. doi: 10.1109/TVCG.2013.212

The paper describes an analytical workflow for interactively improving topic models of document collections. Its central research question is how visual analytics can make topic modelling more reliable, interpretable, and steerable than common probabilistic approaches such as LDA. The workflow begins with a document corpus represented as a term-document matrix. Instead of applying LDA, the system uses nonnegative matrix factorization, which decomposes the corpus into two interpretable parts: topics as weighted keyword combinations and documents as weighted topic combinations. This first model provides an initial analytical overview of the corpus.

The initial result is then visualized so that the analyst can inspect both semantic and structural aspects of the model. Documents are shown in a scatterplot layout using a modified t-SNE projection, with points colored by their strongest topic membership. Topic summaries are shown through representative keywords, and individual documents can be read with topic keywords highlighted. This allows the researcher to ask whether the discovered topics correspond to meaningful concepts, whether clusters are too broad or too narrow, and whether documents appear in plausible topic groups.

The main analytical step is human steering. When the analyst detects a problem in the topic structure, they express their domain knowledge through direct interactions. If two topics appear semantically redundant, they can be merged. If one topic contains multiple concepts, it can be split. If an interesting document or set of documents suggests a missing theme, the analyst can create a new document-induced topic. If a set of keywords represents a concept that should be present, a keyword-induced topic can be created. The analyst can also directly refine keyword weights in an existing topic, for example by removing generic terms or strengthening more meaningful ones.

These interactions are not merely visual edits. They are translated into semi-supervised NMF constraints, meaning that the topic model is recomputed while being guided by the user’s feedback. The workflow therefore alternates between modelling, visualization, interpretation, intervention, and recomputation. Because NMF behaves more deterministically and converges more stably than LDA in the authors’ experiments, the analyst can treat each intervention as a controlled refinement rather than as a restart producing an unrelated model. Intermediate algorithm outputs are visualized during computation, allowing the analyst to see how documents move between topics and how the model evolves.

The analytical logic is therefore iterative and corrective. The system first proposes a topic structure, the analyst evaluates it against semantic expectations and research goals, and the model is then adjusted to better reflect meaningful concepts in the corpus. The paper’s case studies with visualization papers and product reviews show how this workflow supports both broad corpus understanding and targeted discovery, such as isolating negative reviews or distinguishing subtopics in visual analytics research.

10. Progressive Abstraction Analysis of Multivariate Temporal Data

Andrienko, N., Andrienko, G. and Shirato, G. (2023), "Episodes and Topics in Multivariate Temporal Data." *Computer Graphics Forum*, 42: e14926. doi: 10.1111/cgf.14926

This paper presents a workflow for analysing multivariate temporal data by progressively abstracting from raw time series to interpretable patterns of behaviour. The research question is how analysts can understand the distribution of dynamic, multi-attribute characteristics across many episodes, where an episode is a bounded time interval describing part of a process or behaviour. The key difficulty is that each episode contains multiple time series, one per attribute, so direct visual inspection quickly becomes cognitively overwhelming. The workflow addresses this by transforming temporal variation into symbolic patterns and then using topic modelling to discover recurring combinations of those patterns.

The workflow begins by defining or extracting episodes. These may come from a sliding time window, as in the COVID-19 mobility case study, or from domain events, as in the football case study where episodes correspond to periods around changes in ball possession. Once the episodes are established, the first abstraction step converts each single-attribute time series within an episode into a compact symbolic representation. In the authors' implementation, this is done using SAX, which divides a time series into segments and encodes value levels with symbols. The result is that the variation of one attribute during one episode becomes a "word."

The second abstraction step treats each episode as a "text" composed of such symbolic words, one or more for each attribute. Topic modelling is then applied to the collection of episode-texts. In this setting, a topic is not a theme in natural language but a recurring pattern of co-occurring single-attribute temporal behaviours. For example, in mobility data, a topic may combine reduced visits to workplaces and transit stations with increased time at home. In football data, a topic may combine increasing defensive pressure, changing team shape, and ball movement toward or away from goal. Thus, topic modelling becomes a method for discovering multi-attribute temporal patterns.

The third step is interpretation and contextual exploration. Analysts inspect the symbolic patterns belonging to each topic, assign semantic meaning to the topics, and examine how strongly topics occur in each episode. Visualizations are chosen according to the context of the data. In the COVID-19 case, matrix displays and small multiples show topic distributions across countries and time, enabling analysts to relate mobility patterns to mortality waves and lockdown periods. In the football case, topic weights are mapped onto pitch diagrams, allowing analysts to interpret topics as forms of attacking and defensive behaviour.

The workflow is explicitly iterative. Analysts may adjust symbolic encodings, compare topic modelling methods such as LDA and NMF, merge semantically similar topics, or reconsider preprocessing choices if topics are unclear. The analytical logic is progressive: raw values are abstracted into single-attribute patterns, these are abstracted into multi-attribute topics, and topic distributions are then analysed over time, space, and context. This directly

serves the paper’s aim of making complex temporal phenomena understandable at multiple levels of abstraction.

11. Partition-based Regression Modelling

T. Mühlbacher and H. Piringer, “A Partition-Based Framework for Building and Validating Regression Models,” *IEEE Transactions on Visualization and Computer Graphics*, 19(12), pp. 1962–1971, Dec. 2013. doi: 10.1109/TVCG.2013.125

This paper describes an interactive workflow for building, validating, and comparing regression models by visually analysing how a quantitative target varies over features and feature pairs. The research question is how analysts can incorporate domain knowledge into regression modelling, especially when relationships are nonlinear, local, or involve interactions between variables. Instead of relying only on automatic feature selection or global statistical measures, the workflow lets analysts inspect local conditional distributions of the target across partitioned feature spaces.

The workflow begins with a dependent variable, such as natural gas consumption, and a set of candidate independent variables, such as temperature, wind speed, hour, or day of year. Each feature, and each pair of features, is partitioned into disjoint regions. These partitions may be uniform in the feature domain or uniform in data frequency. For each region, the system summarizes the local distribution of the target, for example by showing medians, quartiles, means, or dispersion. This produces a visual overview of how the target changes locally across one-dimensional and two-dimensional feature spaces.

The next step is ranking. Because inspecting every feature and feature pair is impractical in high-dimensional data, the system quantifies relevance using partition-based regression models. It fits simple local models over partitions and ranks variables or variable pairs by how well they explain the target. The ranking guides the analyst toward potentially important features, while the visualizations reveal the structure behind the ranking. This combination is essential: the ranking indicates where to look, but the visualization helps determine whether the relationship is meaningful, nonlinear, noisy, local, or domain-plausible.

For model building, the workflow proceeds iteratively. The analyst first examines the raw dependent variable against candidate features to identify an initial explanatory feature or transformation. A first regression model is then fit. The target of analysis is then changed from the original variable to the model residuals. Visualizing residuals over features and feature pairs reveals what structure remains unexplained. If residuals show systematic patterns over another feature, interaction, or time period, the analyst adds that feature or transformation to the model. The model is refit, residuals are recalculated, and the process repeats.

The same framework also supports validation and comparison. Residual visualizations show where a model overestimates or underestimates. Differences between residual magnitudes of two models show local superiority of one model over another. Variance across predictions from model ensembles can reveal local uncertainty. In the gas-consumption case study, this logic leads from a temperature-based model to a richer model incorporating hour, seasonal effects, weekend effects, and interactions with wind

speed. The analytical workflow therefore moves from broad relevance discovery to residual-driven refinement and finally to model comparison. It connects directly to the paper's goal: making regression modelling a human-guided process in which visual evidence and domain reasoning shape feature selection, transformation, and validation.

12. Spatio-temporal Analysis and Modelling

N. Andrienko and G. Andrienko, "A visual analytics framework for spatio-temporal analysis and modelling," *Data Mining and Knowledge Discovery*, 27(1), 2013, pp. 55–83. doi: 10.1007/s10618-012-0285-7

This paper presents a workflow for analysing and modelling large collections of spatially referenced time series. The research question is how visual analytics can help analysts move from visually perceived spatio-temporal patterns to explicit, reusable, computer-processable models. The workflow is designed for data in which many places, regions, objects, or flows each have their own time series, such as mobile phone activity in city cells or traffic flows between areas.

The workflow begins with data preparation. Raw spatio-temporal events or movement records are transformed into spatial time series through aggregation. For example, mobile phone call records are aggregated into hourly call counts per cellular network region, and vehicle trajectories are aggregated into hourly movement counts and average speeds between neighbouring spatial cells. The analyst may also transform the values, for example normalizing or deriving dependency series, depending on the modelling goal.

The next step is grouping. Rather than modelling every spatial time series independently, the analyst groups time series with similar temporal behaviour. This is done using clustering methods such as k-means or self-organizing maps, combined with interactive refinement. The time graph shows the temporal profiles of each group, while the map shows the spatial distribution of the groups. The analyst uses both displays to judge whether the grouping is internally coherent and spatially meaningful. If a group is too heterogeneous, it can be subdivided; if groups are too similar, they can be merged. This step captures the spatial component of the phenomenon by identifying regions or objects that behave similarly over time.

Once groups are established, the analyst models the temporal behaviour of each group. A representative time series is derived for a group, for example using means, medians, or percentiles at each time step. The analyst visually inspects this representative series to determine whether it contains cycles, trends, or other temporal patterns. They then choose an appropriate time-series modelling method, such as exponential smoothing, and adjust model parameters interactively. The model's predicted curve is shown together with the original and representative time series so that the analyst can judge whether it captures the essential pattern without overfitting local fluctuations.

The following step is model evaluation. Residuals, meaning differences between observed and model-predicted values, are visualized over time and space. If the residuals look random, the model is considered to capture the relevant temporal structure. If residuals show temporal cycles, spatial concentration, or systematic deviations, the analyst returns

to grouping or modelling and refines the analysis. This iterative residual inspection is central because it links model quality to interpretable spatio-temporal patterns.

Finally, the resulting models are stored externally in a machine-readable form, together with group membership and statistics needed for prediction. The models can then be reused to forecast future values, detect anomalies, or communicate findings. The analytical logic is therefore a decomposition of spatio-temporal modelling: first represent spatial variation through groups of similar time series, then represent temporal variation within each group through formal statistical models, and finally evaluate whether the combination adequately explains the observed phenomenon.

13. Feature Engineering for Behaviour Pattern Recognition

N. Andrienko, G. Andrienko, A. Artikis, P. Mantenoglou and S. Rinzivillo, “Human-in-the-Loop: Visual Analytics for Building Models Recognizing Behavioral Patterns in Time Series,” IEEE Computer Graphics and Applications, 44(3), pp. 14–29, May–June 2024. doi: 10.1109/MCG.2024.3379851

This paper describes a human-in-the-loop workflow for creating training data and features for machine learning models that recognize behavioural patterns in time series. The research question is how analysts can build flexible pattern-recognition models when expert concepts are vague, formal rule definitions are too rigid, and labelled training examples are scarce. The workflow uses visual analytics to translate domain knowledge into interval-based features, identify representative examples, and prepare data for machine learning.

The workflow starts by converting raw time series into episodes. Since behavioural patterns occur over intervals rather than at individual time points, the analyst first segments trajectories or multivariate time series into intervals of suitable duration. In the vessel case study, long fishing-vessel trajectories are cut into overlapping three-hour windows so that trawling behaviour is likely to be captured. In the football case study, episodes correspond to periods of ball possession. This step reframes the data from point-level measurements to candidate behavioural units.

The second step is feature engineering. For each episode, the analyst derives interval-based features that summarize relevant aspects of behaviour. These may describe value levels, variability, trends, curvature, deviations, or fluctuations. The choice of features is guided by domain knowledge. For trawling, relevant characteristics include low speed and repeated changes in direction, so the authors derive smoothed speed measures, curvature of movement, and distance to ports. For football attacks, features describe the shape of ball movement, such as lateral range, forward progression, and deviation from a trend line. The point is not to impose a rigid formal definition but to construct features that make relevant behavioural differences visible and learnable.

The third step assesses whether the features discriminate meaningful patterns. Analysts inspect feature distributions, use projections such as UMAP, and examine groups of similar episodes in domain-appropriate visualizations, such as maps for vessel trajectories or pitch diagrams for football. If a group contains mixed behaviours, the feature set is inadequate

and the analyst returns to feature construction or selection. If groups are coherent, the features are considered useful.

The fourth step creates labelled examples. Rather than labelling episodes one by one, the analyst selects groups or cluster cores of similar episodes and labels them as pattern examples. This is efficient because the visualization allows many examples to be judged together. Importantly, the analyst labels both positive examples, such as trawling, and negative or alternative behaviours, such as port entry or different football attack styles. The workflow can also reveal unexpected subtypes, such as multiple forms of trawling trajectories or several variants of possession play.

Finally, the suitability of the produced training data is evaluated using a simple classifier such as k-nearest neighbours. If unlabelled episodes similar to the labelled examples are classified plausibly, the feature set and training examples are judged suitable for training a more advanced model. The analytical logic is therefore iterative: segment behaviour, construct interval features, visually test their meaning, label representative groups, and validate whether similarity in feature space corresponds to similarity in behavioural meaning.

14. Exploratory Model Analysis

Cashman, D., Humayoun, S.R., Heimerl, F., Park, K., Das, S., Thompson, J., Saket, B., Mosca, A., Stasko, J., Endert, A., Gleicher, M. and Chang, R. (2019), "A User-based Visual Analytics Workflow for Exploratory Model Analysis." *Computer Graphics Forum*, 38: 185–199. doi: 10.1111/cgf.13681

This paper defines and operationalizes a workflow for exploratory model analysis, where the goal is not primarily to understand a dataset but to discover what useful predictive models can be built from it. The research question is how visual analytics can support users when the modelling task itself is not known in advance. Unlike systems that assume a predefined model type or target variable, this workflow helps users explore possible prediction problems, generate candidate models, compare them, and export models suitable for later use.

The workflow begins with data exploration. The system provides visual summaries of the input data, such as histograms for tabular attributes, node-link diagrams for graph data, time-series plots, or other views depending on the data type. This step helps the user understand variables, distributions, and possible relationships before committing to a modelling task. The data are not only inspected for insight, but also to judge what might be predictable and useful.

The second step is problem exploration. Based on the dataset, the system automatically proposes possible modelling problems. A problem specification includes the target variable, the type of model or task, the predictor variables, and the performance metric. For example, categorical variables may produce classification problems, numeric variables may produce regression problems, and graph data may generate graph-specific tasks. The user reviews these suggestions in light of their goals and understanding of the data.

The third step is problem specification refinement. The user selects or creates a modelling problem and adjusts its details. They may remove irrelevant or inappropriate features, choose a performance metric, or redefine the target. This step bridges exploratory data analysis and machine learning: the user turns an emerging analytical question into a machine-readable modelling task.

The fourth step is model generation. An automated machine learning backend receives the problem specification and training data, then produces multiple candidate models. These models may differ in algorithm, feature processing, hyperparameters, and structure. The purpose is not to optimize one known model but to populate a model space that the user can inspect.

The fifth step is model exploration. Candidate models are visualized through their predictions on holdout data. Classification models may be shown as confusion matrices; regression models as residual plots; time-series forecasting models as predicted versus observed curves. These views are linked back to the data, allowing the user to understand where models succeed or fail and whether their errors matter for the application context.

The sixth and seventh steps are model selection and export. The user selects one or more preferred models, not necessarily the one with the best aggregate score. For example, a model with slightly lower accuracy may be preferable if it performs more evenly across classes or has fewer dangerous error types. The selected model is then exported for deployment or further use. The workflow can loop back to problem specification at any point, allowing users to explore different modelling goals on the same dataset.

The analytical logic is therefore model-centred rather than data-centred: understand the data enough to pose candidate prediction problems, use automation to generate model alternatives, use visualization to compare model behaviour, and select deployable models based on both metrics and contextual judgement.

15. Diagnosing Binary Classifiers

J. Krause, A. Dasgupta, J. Swartz, Y. Aphinyanaphongs and E. Bertini, "A Workflow for Visual Diagnostics of Binary Classifiers using Instance-Level Explanations," 2017 IEEE Conference on Visual Analytics Science and Technology (VAST), Phoenix, AZ, USA, 2017, pp. 162–172. doi: 10.1109/VAST.2017.8585720

This paper presents a workflow for diagnosing binary classifiers by combining aggregate performance summaries with instance-level explanations. The research question is how data scientists and domain experts can understand, validate, and improve black-box classifiers, especially in high-stakes domains such as healthcare. The workflow is designed not merely to report accuracy but to support semantic validation: experts need to judge whether the model's decisions make sense in terms of domain knowledge.

The workflow begins after a binary classifier has already been trained. The system first computes an explanation for each instance. An explanation is the minimal set of features whose removal would change the classifier's predicted label. In the paper's healthcare case, where the model predicts whether emergency-room patients will be admitted based on medications, an explanation identifies which drugs drive a particular prediction. Because

this method treats the classifier as a black box, the workflow can be applied to different model types.

The first visual step is an outcome-level statistical summary. The user inspects score distributions, a confusion matrix, and an ROC curve. This establishes the model's overall performance and shows how correct and incorrect predictions are distributed across prediction scores. The analyst can identify whether errors cluster near the decision threshold or whether the model makes confident mistakes. This step answers the broad question of how accurate and reliable the classifier appears.

The second step moves from outcomes to explanations. Instances are grouped by shared explanations, and each explanation group is shown with its feature set, number of instances, distribution of true and false positives or negatives, and odds ratio. This view reveals the main "decisions" the model makes. For example, if many patients are explained by Sodium Chloride, the analyst can see whether that explanation strongly predicts admission or whether it is ambiguous. Sorting and filtering allow the user to focus on common explanations, highly predictive explanations, uncertain explanations, or explanations involving specific features.

The third step drills down to the raw data for a selected explanation. The Item Level Inspector displays the feature vectors of the instances in that explanation group, allowing the analyst to compare correctly and incorrectly classified cases. This helps identify why a decision pattern fails. In the case study, some medications were found to be weak or ambiguous signals because they were used in many contexts, while certain diagnostic features improved prediction but also introduced label leakage because they were recorded too close to the admission decision.

The final analytical move is outside the visualization but enabled by it: the analyst uses these insights to revise the data, feature set, or model design. The workflow can reveal whether poor performance is due to model choice, missing information, ambiguous features, or limitations of the prediction task itself. Its logic is therefore diagnostic and iterative: summarize performance, identify decision patterns through explanations, inspect raw instances behind those patterns, infer root causes, and then return to modelling with better assumptions or data.

16. Interactive Exploration of Trained Ensemble Classifier

Eirich, J., Münch, M., Jäckle, D., Sedlmair, M., Bonart, J. and Schreck, T. (2022), "RfX: A Design Study for the Interactive Exploration of a Random Forest to Enhance Testing Procedures for Electrical Engines." *Computer Graphics Forum*, 41: 302–315. doi: 10.1111/cgf.14452

This paper describes a workflow for using visual analytics to make a trained Random Forest understandable and actionable for automotive engineers. The research question is how domain experts without advanced machine-learning expertise can derive practical testing rules from a Random Forest classifier. The application context is quality control for electrical engines, where many sensor signals are recorded during test procedures and engineers need to identify combinations of measurements that reveal faulty components.

The workflow begins with a Random Forest trained on extracted features from sensor measurements and binary labels indicating whether a component is acceptable or faulty. The forest contains many decision trees, each representing a possible set of measurement-based decision rules. Since inspecting every tree individually is impractical, the first analytical step is to compute similarities between trees. The authors combine semantic similarity, based on whether trees make similar predictions on test samples, with structural similarity, based on the similarity of tree paths and features. This produces a dissimilarity matrix that is used to cluster and project trees into a two-dimensional overview.

The user begins analysis in the Random Forest overview. Each decision tree appears as a point, with similar trees positioned near each other and clusters marked visually. The purpose is to partition the forest into regions of similar decision logic. The analyst then identifies clusters that look promising, for example because they have high accuracy or contain compact groups of similar trees. This step turns a large ensemble into a smaller set of candidate regions for inspection.

The next step is to identify representative trees within these clusters. The system shows candidate trees using icicle plots and table views, allowing the user to compare tree accuracy, complexity, and decision-path structure. The analyst selects trees that appear both accurate and interpretable, often preferring trees with fewer nodes because they can more easily be converted into testing rules. This step balances predictive performance with practical usability.

Once a tree is selected, the workflow moves to detailed exploration. The tree is displayed as a node-link diagram in which each node contains a confusion-matrix-like summary of class distribution and prediction quality. Edges show split thresholds, and node labels identify the measurement features used in decisions. By following paths through the tree, the engineer can interpret the model as a sequence of rules, such as testing one signal threshold and then evaluating another measurement if the condition holds.

The final step is rule refinement and validation. Selecting a node opens a histogram of the underlying feature distribution, separated by class. The analyst can adjust split thresholds interactively using domain knowledge and immediately see how this changes classification behaviour within the tree. Candidate rules are then tested on new sensor data before being incorporated into real testing procedures. In the reported field study, rules derived from RfX helped detect rotor eccentricity earlier in the testing process, reducing testing time for a substantial subset of faulty components.

The analytical logic is therefore to transform an opaque ensemble into interpretable, testable rules: cluster similar trees, select representative and accurate trees, inspect their decision paths, refine thresholds with domain knowledge, and validate the resulting rules in the manufacturing process.

17. Exploring Deep Learning Models in TensorFlow

Kanit Wongsuphasawat, Daniel Smilkov, James Wexler, Jimbo Wilson, Dandelion Mané, Doug Fritz, Dilip Krishnan, Fernanda B. Viégas, and Martin Wattenberg, "Visualizing Dataflow Graphs of Deep Learning Models in TensorFlow," IEEE Transactions on

The analytical workflow in this paper is designed to help machine-learning developers understand, debug, and communicate the structure of TensorFlow models. The research problem is that TensorFlow models are represented internally as large, low-level dataflow graphs, often containing thousands of operations, but developers reason about them as higher-level neural network architectures. The workflow therefore begins with a raw TensorFlow computation graph and transforms it into an interactive visual representation that better matches the conceptual model developers use when designing and discussing deep learning systems.

The first analytical step is simplification. The raw graph contains many nodes that are technically necessary but analytically distracting, such as constants, summaries, and bookkeeping operations. The workflow identifies such non-critical nodes and removes them from the main layout while preserving their relationship to relevant operations through compact side encodings. This step matters because the paper’s central question is not how to visualize every low-level operation equally, but how to expose the meaningful architecture of the model. By reducing visual clutter, the workflow allows the main computational structure to emerge.

The second step is abstraction through hierarchy. TensorFlow operation names often include namespaces, which reflect how developers organize code into conceptual modules. The visualizer uses these namespaces to cluster operations into nested groups. This creates an overview of the model at the level of major components, such as input processing, convolutional layers, training operations, and loss computation. The analyst can then expand groups on demand to inspect their internal operations. Thus, the workflow moves from global understanding to local detail: first seeing the model’s high-level dataflow, then drilling down where debugging or explanation requires it.

The third step is stabilizing exploration. Because expanding graph clusters can otherwise cause large layout changes, the workflow routes and bundles edges through the hierarchy. This preserves the user’s mental map while exploring nested structure. The workflow also extracts auxiliary high-degree nodes, such as initialization, saving, or training-related hubs, to the side of the main graph. These nodes remain accessible, but they no longer dominate the layout. This lets developers distinguish the core inference architecture from support machinery.

After the graph is made readable, the workflow supports analytical interpretation. Repeated structures are detected and highlighted with consistent colors, helping users recognize modular design patterns such as repeated convolutional or inception modules. Quantitative overlays, such as tensor shape, memory use, compute time, or device placement, can then be added to connect structural understanding with performance analysis. A developer may begin with an overview, notice repeated layers, expand one layer to inspect implementation details, check whether tensors flow as intended, and then use runtime overlays to identify bottlenecks.

Overall, the workflow answers the paper’s research question by turning an opaque low-level computation graph into a navigable analytical artifact. It supports a progression from

decluttering, to architectural overview, to interactive inspection, to quantitative debugging, aligning the visualization with the way researchers and engineers reason about deep learning models.

18. What-If Probing of ML Models

James Wexler, Mahima Pushkarna, Tolga Bolukbasi, Martin Wattenberg, Fernanda B. Viégas, and Jimbo Wilson, “The What-If Tool: Interactive Probing of Machine Learning Models,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 26, pp. 56–65, 2019. doi: 10.1109/TVCG.2019.2934619

The analytical workflow in the What-If Tool is built around interactive probing of trained machine-learning models. The paper’s research question is how practitioners can understand model behavior, compare models, and investigate fairness without writing extensive custom code or inspecting model internals. The workflow therefore treats the model largely as a black box and focuses on the relationship between input examples and model outputs.

The workflow begins by loading a trained model, or two models for comparison, together with a sample dataset. The first analytical stage is broad data and prediction exploration. Users inspect the dataset through configurable visualizations, such as scatterplots, histograms, confusion matrices, small multiples, or image thumbnails. These views can encode both input features and model outputs, allowing users to see where predictions are correct, incorrect, uncertain, or different between models. This stage establishes the global behavioral landscape of the model: how predictions vary across the data and whether obvious subgroups or anomalies deserve closer inspection.

The next stage moves from global inspection to local hypothesis testing. A user selects an individual data point and examines its feature values and model prediction. The workflow then allows direct editing of those feature values and re-running inference. This supports “what-if” reasoning: for example, asking how a loan decision, income prediction, or image classification would change if one feature were altered. The analytical logic is counterfactual and experimental. Rather than simply accepting a prediction, the user perturbs inputs to reveal which changes matter to the model.

The workflow then broadens this local probing through automatic counterfactual search and partial dependence analysis. Counterfactual examples identify similar data points that receive different predictions, helping users ask what minimal differences might explain a changed outcome. Partial dependence plots show how predictions change as one feature varies across a range of values. Together, these steps help users connect individual cases to broader feature-response patterns. They also reveal unintuitive model behavior, such as a model ignoring a feature, reacting non-monotonically to a feature, or treating correlated features differently.

A further stage focuses on performance and fairness. Users define a ground-truth label and slice the dataset by one or more features, including intersectional subgroups such as race and gender or age and sex. The tool then computes performance metrics such as accuracy, false positive rate, false negative rate, confusion matrices, ROC curves, or regression errors. For binary classifiers, users can adjust classification thresholds, cost

ratios, and fairness optimization strategies such as demographic parity, equal opportunity, or equal accuracy. This lets analysts test how model behavior changes under different operational definitions of fairness.

Finally, the workflow supports comparison between models. Users can inspect where two models agree or disagree, compare partial dependence patterns, and evaluate whether a revised model improves performance or fairness. Across the workflow, the analytical logic is iterative: visualize, select, perturb, compare, slice, adjust thresholds, and repeat. The paper’s contribution is a workflow that makes model understanding an exploratory process, linking individual explanations, global behavior, subgroup performance, fairness metrics, and model comparison in one interactive loop.

19. Active Gaze Labeling: Visualization for Trust Building

Maurice Koch, Nan Cao, Daniel Weiskopf, and Kuno Kurzhals, “Active Gaze Labeling: Visualization for Trust Building,” IEEE Transactions on Visualization and Computer Graphics, vol. 31, no. 7, pp. 3942–3956, July 2025. doi: 10.1109/TVCG.2024.3392476

The workflow in this paper addresses the problem of labeling areas of interest, or AOIs, in eye-tracking video data. The research question is how visualization can support semi-automatic gaze labeling while exposing classifier uncertainty, helping users detect ambiguous or erroneous labels, and building trust in the classifier over time. The workflow combines human annotation, dimensionality reduction, uncertainty visualization, and active learning.

The process begins with raw eye-tracking recordings consisting of gaze points over video stimuli. These gaze samples are aggregated into fixations, and for each fixation the system extracts image thumbnails representing what the participant was looking at. These fixation thumbnails become the basic analytical objects. Because the aim is to label gaze behavior semantically, the workflow does not ask the user to annotate every video frame directly. Instead, it projects fixation thumbnails into a two-dimensional embedding, where visually similar fixations are placed near each other. This produces an initial overview of the gaze data and allows the annotator to identify clusters that likely correspond to AOIs.

The first analytical stage is initial labeling. The user inspects clusters in the embedding, selects fixations that appear to represent the same AOI, and assigns labels. At this stage, visualization supports the human in discovering the relevant AOI categories and creating an initial training set. The workflow then trains a classifier on these manually labeled examples. This begins the active learning loop: the classifier predicts labels for new or unlabeled fixations, and the visualization helps the user decide which predictions to accept, correct, or inspect more carefully.

A key part of the workflow is uncertainty-aware analysis. The EyeFlower glyph represents each fixation with the fixation thumbnail at the center and petals showing the classifier’s probability distribution over AOI labels. This makes uncertain or conflicting predictions visible at the individual-fixation level. At the aggregate level, a multi-class heatmap shows where labeled classes are concentrated in the embedding, while overlap indicators highlight areas where multiple label densities intersect. These overlapping regions are

analytically important because they often correspond to ambiguous fixations, labeling errors, or semantic boundary cases.

The workflow also supports retrospective analysis and outlier detection. An interactive lens lets the user inspect how fixation samples relate to the embedding, revealing whether samples from a fixation are tightly contained within one AOI region or spread across several. This is crucial because gaze is spatially uncertain: a fixation may lie between two objects, or its samples may cover multiple AOIs. The user can use this evidence to detect labels that are inconsistent, ambiguous, or difficult for the classifier to learn. Corrected labels are then fed back into the classifier, improving subsequent predictions.

Over time, the user's role shifts from manually labeling many fixations to supervising and validating the classifier. Trust is built not by hiding uncertainty, but by making the classifier's confidence, failures, and edge cases visible. The workflow therefore answers the paper's research question by integrating labeling, uncertainty inspection, classifier refinement, and explanation into a single iterative process. Its analytical logic is to start with human-understandable clusters, train a model, expose uncertainty, focus human attention on problematic cases, and progressively increase confidence in automatic AOI labeling.

20. COQUITO Workflow

Josua Krause, Adam Perer, and Harry Stavropoulos, "Supporting Iterative Cohort Construction with Visual Temporal Queries," *IEEE Transactions on Visualization and Computer Graphics*, vol. 22, no. 1, pp. 91–100, Jan. 2016. doi: 10.1109/TVCG.2015.2467622

The COQUITO workflow supports researchers who need to construct cohorts from large event databases using temporal constraints. The paper's research question is how domain experts, such as medical or social media researchers, can define complex temporal cohorts without writing SQL or relying on database specialists. The workflow is built around visual temporal querying, where cohort construction is treated as an iterative analytical process rather than a one-time database extraction.

The workflow begins with the full population represented as the source of a visual query graph. The researcher then adds constraints by dragging event types into the graph. Each event constraint represents something that must occur in an individual's timeline, such as a diagnosis, medication, procedure, or social media check-in. As constraints are added, the system creates junctions that represent intermediate cohort states. Each junction shows how many individuals satisfy the query up to that point, so the researcher receives immediate feedback about whether the cohort is becoming too broad or too narrow.

The next analytical step is specifying temporal relationships. Events can be arranged in sequence, constrained to occur within time windows, or combined conjunctively and disjunctively. For example, a medical researcher might define patients who receive a diagnosis, then undergo one of several treatments within a specified number of days. Branching paths encode alternatives, while multiple constraints in a junction encode events that must jointly occur. Negation and demographic filters allow the researcher to exclude populations or restrict the cohort by attributes such as age, gender, or ethnicity. The

analytical logic is incremental: each added constraint tests a hypothesis about what temporal pattern defines the cohort.

COQUITO also supports exploration while constructing the query. Search tools help locate relevant event types, while treemaps summarize which events commonly occur after a selected query junction. This helps users decide what constraints might be meaningful to add next. Demographic summaries show the composition of the current cohort at any stage, allowing the researcher to detect unintended biases or surprising shifts in the population. Thus, query construction is not separated from data understanding; the user learns about the dataset while defining the cohort.

A further stage supports comparison and reuse of cohorts. Users can reference the result of one query in another query, negate it, or use it to create matched case and control cohorts. This is especially important in medical predictive modeling, where researchers often need one cohort with a condition and another comparable cohort without it. Matching and export functions allow the visually defined cohorts to be passed into downstream statistical or predictive modeling tools.

The workflow directly addresses the paper’s motivating problem: temporal cohort definitions are often complex, iterative, and difficult to translate into database queries. COQUITO turns cohort construction into a visual reasoning process. Researchers begin with the entire population, progressively apply temporal and attribute constraints, monitor intermediate results, refine the query based on feedback, and finally export or compare the resulting cohorts. The workflow’s value lies in making cohort definition itself an interactive analytical activity, enabling domain experts to explore and operationalize temporal hypotheses without losing sight of cohort size, composition, or downstream research goals.

21. Visual Cohort Comparison for Spatial Single-Cell Omics-Data

Antonios Somarakis, Marieke E. Ijsselsteijn, Sietse J. Luk, Boyd Kenkhuis, Noel F.C.C. de Miranda, Boudewijn P.F. Lelieveldt, and Thomas Höllt, “Visual Cohort Comparison for Spatial Single-Cell Omics-Data,” IEEE Transactions on Visualization and Computer Graphics, vol. 27, no. 2, pp. 733–743, Feb. 2021. doi: 10.1109/TVCG.2020.3030336

The workflow in this paper supports clinical researchers comparing two cohorts of spatially resolved single-cell omics samples. The research question is how visual analytics can help identify biological features that differentiate cohorts when both cell-type abundance and spatial cellular organization may matter. The workflow therefore proceeds from coarse quantitative comparison to increasingly detailed spatial and tissue-level inspection.

The process begins with preprocessed tissue images in which cells have already been segmented and assigned cell-type labels. Each cohort contains multiple samples, and each sample contains many cells. The first analytical stage compares the cohorts by cell-type abundance. For every cell type, the workflow computes how frequently that cell type appears in each sample and visualizes the distribution across the two cohorts using superposed raincloud plots. These plots simultaneously show cohort-level distributions and individual sample positions. This lets the researcher see whether a cell type is more

common in one cohort, whether the cohorts overlap, and whether individual samples behave as outliers.

The workflow supports exploration of many cell types by allowing sorting according to how strongly a cell type separates the cohorts, textual search for known biological categories, and drag-and-drop aggregation of related cell types. This is important because clinically meaningful categories may be distributed across several fine-grained cell labels. For example, multiple T-cell subtypes may need to be combined to assess immune infiltration. Once a potentially differentiating cell population is identified, the researcher can select samples or outliers and inspect their spatial locations in the tissue view.

The second analytical stage moves from abundance to spatial organization. The workflow defines each cell's microenvironment as the nearby cells within a user-defined distance. It then compares how often pairs of cell types co-occur in microenvironments across the two cohorts. A difference heatmap highlights pairwise cell-type interactions that are more abundant in one cohort than the other. This provides a high-level overview of spatial co-localization patterns, helping researchers identify candidate interactions that may explain functional differences between cohorts.

The third stage allows detailed microenvironment construction. Starting from a pairwise interaction selected in the heatmap, the researcher can build more complex microenvironment patterns by adding cell types through an interactive query-like interface. For each constructed microenvironment, raincloud plots again show its abundance across samples in both cohorts. This lets the analyst determine whether a spatial pattern, rather than a single cell type alone, differentiates the cohorts. Outliers can be detected at this stage as well, such as samples whose microenvironment abundance is unexpectedly high or low relative to their cohort.

Throughout the workflow, abstract findings are linked back to the original tissue images. Selected cell types, samples, or microenvironments are highlighted in their spatial context. This final step is essential because preprocessing uncertainty, segmentation errors, and biological plausibility must be checked visually. The researcher can verify whether an identified pattern corresponds to meaningful tissue structure or an artifact.

Overall, the workflow answers the paper's research question by organizing cohort comparison into a layered analytical process: compare cell abundance, detect outliers, compare pairwise spatial interactions, build detailed microenvironment hypotheses, and verify findings in tissue. This supports exploratory biomarker discovery in spatial single-cell data, where differentiating features may lie not only in which cells are present, but also in how they are spatially arranged.

22. Coral Workflow

Patrick Adelberger, Klaus Eckelt, Markus J. Bauer, Marc Streit, Christian Haslinger, and Thomas Zichner, "Coral: a Web-Based Visual Analysis Tool for Creating and Characterizing Cohorts," *Bioinformatics*, vol. 37, no. 23, pp. 4559–4561, 2021. doi: 10.1093/bioinformatics/btab695

The Coral workflow is designed for interactive cohort creation and characterization in cancer genomics. The paper's research question is how analysts can define, refine, compare, and reproduce multiple patient or sample cohorts based on many genomic and metadata attributes. Coral treats cohort analysis as an iterative process in which each filtering, splitting, comparison, or inspection step becomes part of a visible analysis history.

The workflow begins with a loaded dataset, such as AACR Project GENIE, TCGA, or CCLE. Coral automatically creates an initial cohort containing all items in the dataset. This cohort appears in the Cohort Evolution View, a graph that records how cohorts are generated and related. The analyst selects one or more cohorts from this graph, and those cohorts are then loaded into the Action View for analysis. This establishes the core logic of the workflow: cohorts are not isolated query results, but evolving objects whose provenance is visually preserved.

The next step is cohort creation through filtering and splitting. Filtering creates a new cohort by selecting items that satisfy a condition, such as having a particular KRAS mutation, tumor type, sex, race, or other molecular or clinical attribute. Splitting divides a cohort into multiple subcohorts based on the values of an attribute, such as race or cancer type. These operations can be applied repeatedly. Each resulting cohort is added back to the evolution graph, so the analyst can always see which operations produced which groups. This supports complex cohort definitions while maintaining interpretability and reproducibility.

Once cohorts exist, the workflow moves to characterization. The View operation allows the analyst to inspect how one or more attributes are distributed across selected cohorts. Coral chooses visual encodings based on the number and type of attributes being examined, allowing users to compare metadata, mutation status, expression values, or other features. This step helps answer questions such as whether a biomarker is enriched in a subgroup, whether demographic variables differ across cohorts, or whether a cohort has unexpected internal structure.

The workflow then supports formal comparison. The Compare operation tests selected cohorts pairwise for statistically significant differences and summarizes results in a matrix. Visualizations and test details can be inspected on demand. This step connects exploratory visual patterns to statistical confirmation, helping analysts distinguish apparent differences from robust ones.

A further analytical step is prevalence estimation. After defining a sample cohort with a characteristic of interest, such as KRASG12C mutation, the analyst can flexibly define the reference cohort by applying or skipping prior operations. Coral then computes and displays the resulting prevalence. This is important in cancer genomics, where researchers often need to understand how common a mutation or marker is within clinically meaningful subpopulations.

Finally, the workflow supports drill-down to individual items. Using tabular inspection, analysts can view samples or patients directly, select attributes, sort, filter, and examine outliers. Session management allows findings and cohort definitions to be stored, shared, and reproduced.

Overall, Coral’s workflow answers the paper’s research question by combining cohort construction, provenance tracking, characterization, statistical comparison, prevalence analysis, and item-level inspection. Its analytical logic is iterative: define cohorts, inspect their properties, refine them, compare them, estimate prevalence, and preserve the entire reasoning path for reproducible cancer cohort analysis.

23. DaedalusData Workflow

Alexander Wyss, Gabriela Morgenshtern, Amanda Hirsch-Hüsler, and Jürgen Bernard, “DaedalusData: Exploration, Knowledge Externalization and Labeling of Particles in Medical Manufacturing — A Design Study,” IEEE Transactions on Visualization and Computer Graphics, vol. 31, no. 1, pp. 54–64, Jan. 2025. doi: 10.1109/TVCG.2024.3456329

The DaedalusData workflow supports product quality engineers analyzing particle contamination in medical diagnostics manufacturing. The paper’s research question is how visual analytics can help experts explore thousands of particle images, label them efficiently, and externalize domain knowledge so that contamination assessment becomes more data-driven and reproducible. The workflow is built around the idea that expert labels are not merely annotations, but new knowledge attributes that can reshape later exploration.

The process begins with particle data collected from in-vitro diagnostics consumables. Each particle has an image, image-derived attributes such as size and shape, and production-context attributes such as supplier, lot number, and production date. The first analytical stage is exploratory overview. Experts view particles on a central canvas using either an Attribute View or a Projection View. In the Attribute View, particles are arranged according to one selected attribute, such as lot number or supplier. This allows direct comparison between groups, for example checking whether one production lot contains unusually many blue or reflective particles. In the Projection View, particles are arranged by similarity across multiple selected attributes, helping experts detect clusters, anomalies, and recurring visual patterns.

The next step is narrowing the search space. Experts use filters to restrict the displayed particles by production or image attributes, while preserving context about what has been excluded. They can zoom, pan, adjust particle size, change transparency, and inspect particles in detail. When an interesting group appears, such as a dense cluster of similarly colored particles, the expert selects it using lasso or rectangle tools. A detail view then shows a selected particle’s full attributes, and a selection-inspection view summarizes the selected group’s attribute distributions. This helps the expert interpret whether the group is associated with a particular supplier, lot, size range, or other feature.

The workflow then moves into labeling. Experts create labels, such as “blue,” “bright,” or other domain-relevant categories, and assign them to selected particles. Labels are organized into label alphabets, meaning coherent sets of labels that express a particular expert perspective on the particles. For example, one alphabet might concern color, another material type, and another severity. This step externalizes tacit expert knowledge into structured, reusable data.

The central analytical innovation is that these labels become augmented attributes for subsequent analysis. In a label-informed projection, DaedalusData recomputes particle layouts using both original attributes and expert-provided labels. This allows the system to visually organize particles according to domain semantics, not only measured image features. Experts can then discover unlabeled particles near labeled groups, validate existing labels, identify inconsistencies, and extend the label alphabet. Thus, the workflow becomes cyclical: explore particles, select patterns, label them, project using labels, discover further candidates, and refine the knowledge structure.

The workflow directly supports the paper’s quality-control goal. Engineers can investigate whether a contamination pattern is isolated to one lot, appears across suppliers, or reflects a recurring production issue. The case studies show how experts used the system to confirm anomalous blue particles in one lot, check whether other lots were affected, and use label-informed projections to reveal supplier- or lot-related structures.

Overall, DaedalusData’s workflow turns visual inspection into a systematic knowledge externalization process. It connects image exploration, production context, expert labeling, and semi-supervised projection into a loop that supports more consistent, scalable, and collaborative contamination analysis.

24. Knowledge-Incorporated Embedding Exploration Workflow

Jie Li and Chun-qi Zhou, “Incorporation of Human Knowledge into Data Embeddings to Improve Pattern Significance and Interpretability,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 29, no. 1, pp. 723–733, Jan. 2023. doi: 10.1109/TVCG.2022.3209382

The workflow in this paper addresses a common problem in embedding-based data exploration: projections of high-dimensional data do not always show clear or interpretable visual structures. The research question is how human knowledge can be incorporated into data embeddings so that patterns become more visually significant and easier to explain. The workflow combines knowledge externalization, knowledge-informed embedding, projection exploration, and pattern explanation.

The process begins with a multidimensional dataset and an analyst’s tacit knowledge about meaningful sample groupings. This knowledge may come from domain understanding, such as the idea that countries on the same continent may show similar COVID-19 trends, or that customers with similar income levels may behave similarly. The first analytical stage is knowledge externalization in the Knowledge Editor. The analyst selects an attribute and divides its values into meaningful groups. For categorical attributes, groups may correspond to categories; for numerical attributes, the value range can be binned. These groups become explicit sample labels that represent the analyst’s assumptions about possible structure in the data.

The next stage incorporates these labels into the embedding process. The paper’s embedding network combines two objectives: preserving data-feature similarity and respecting the analyst-defined class information. The reconstruction objective keeps embeddings faithful to the original multidimensional data, while the classification objective pulls samples of the same analyst-defined class closer together when their data features

support that grouping. A classification-loss ratio controls how strongly the human knowledge influences the projection. This step is central to the paper’s research question because it operationalizes human knowledge as a force that can make weak or diffuse patterns more visible without fully overriding the data.

The workflow then moves to projection exploration in the Data Projector. The analyst examines the resulting two-dimensional embedding, where clearer clusters, subclasses, and outliers may emerge. Samples are colored according to the analyst-defined classes, so the user can see whether visual structures are class-consistent. The analyst may adjust the classification-loss ratio to explore different balances between raw data similarity and knowledge-guided structure. Low ratios reveal patterns strongly supported by the data, while higher ratios help expose class structure and potential outliers that resist grouping.

After identifying a cluster, subgroup, or outlier, the workflow proceeds to explanation in the Pattern Explainer. The analyst selects one or two visual structures in the projection and compares them. The system computes which factors best distinguish the selected structures, using explanation scores such as SHAP values. Two kinds of explanations are provided: embedding factors, which explain how the selected structure differs in the original data dimensions, and classification factors, which explain how it relates to the human-defined groups. This connects the visual pattern back to both the data and the analyst’s knowledge.

The workflow is iterative. If the projection reveals unclear or surprising structures, the analyst can return to the Knowledge Editor, refine groups, create subclasses, filter samples, delete classes, or define new knowledge groupings. The embedding and explanation steps are then repeated.

Overall, the workflow answers the paper’s research question by creating a feedback loop between human conceptual knowledge and machine-generated embeddings. Analysts externalize assumptions, the model uses those assumptions to produce more significant projections, the user selects emergent structures, and explanations reveal why those structures formed. The result is an exploratory process in which human knowledge improves pattern visibility while visual analysis tests and refines that knowledge.

25. DPVis workflow

DPVis: Visual Analytics With Hidden Markov Models for Disease Progression Pathways — B. C. Kwon et al., IEEE Transactions on Visualization and Computer Graphics, 27(9), 3685–3700, 2021. doi: 10.1109/TVCG.2020.2985689

The DPVis workflow is designed to help clinical researchers turn longitudinal patient records into interpretable disease progression pathways. The paper’s central research question is how hidden Markov models, which can discover latent disease states from irregular clinical observations, can be made understandable and useful for clinicians who need to reason about disease stages, patient trajectories, and clinically meaningful subgroups. The workflow begins before visualization, when researchers choose the clinical variables that should define disease progression, such as autoantibody measurements in type 1 diabetes. They then train a hidden Markov model or variant, choosing parameters such as the number of latent states and possible transition constraints. The model assigns

each patient visit to a probabilistic latent state, thereby transforming complex multivariate longitudinal records into sequences of inferred disease states.

Once the model has produced states and state sequences, DPVis first supports interpretation of the latent states themselves. Researchers inspect state summaries through visual encodings of variable distributions, asking what each hidden state clinically represents. In the type 1 diabetes example, this means determining whether a state corresponds to no autoantibodies, a first autoantibody, multiple autoantibodies, or a likely pre-diagnosis condition. This step is essential because the later trajectory analysis only makes sense if the states can be mapped to meaningful clinical concepts. DPVis therefore links model output back to the observed clinical variables, including uncertainty and missingness, so that users can evaluate whether the model is clinically plausible.

After interpreting states, the workflow moves to progression pathways. Researchers examine how patients transition between states over age or visits using pathway visualizations such as waterfall-style trajectories, Sankey-like summaries, chord diagrams, and mined frequent transition patterns. This reveals whether the cohort contains distinct progression routes, for example patients who first develop insulin autoantibodies versus those who first develop GADA or multiple autoantibodies. The workflow then becomes interactive and hypothesis-driven: users select pathways, filter by transition timing, or query state sequences to define patient subgroups.

The final analytical stage relates these subgroups to outcomes and patient characteristics. DPVis allows researchers to compare diagnosis age, seroconversion timing, genetic or demographic variables, and individual patient records across trajectory groups. Users can drill down from cohort-level patterns to single subjects, inspect posterior probabilities for uncertain state assignments, and decide whether unusual cases reflect model uncertainty, measurement error, or clinically interesting endotypes. Thus, the workflow proceeds from model construction, to state interpretation, to pathway discovery, to subgroup definition, and finally to clinical association and explanation. Its analytical logic is that HMMs provide a compact progression model, while visual analytics lets clinicians validate, interpret, and use that model for disease-specific reasoning.

26. EnsembleLens workflow

EnsembleLens: Ensemble-Based Visual Exploration of Anomaly Detection Algorithms with Multidimensional Data — K. Xu et al., IEEE Transactions on Visualization and Computer Graphics, 25(1), 109–119, 2019. doi: 10.1109/TVCG.2018.2864825

The EnsembleLens workflow addresses the problem that anomaly detection results depend strongly on the chosen algorithm, parameter setting, and feature subspace, especially when ground truth labels are unavailable. The paper’s research question is how analysts can evaluate and select anomaly detection algorithms for a particular multidimensional dataset when there is no definitive answer about which points are truly anomalous. The workflow therefore treats algorithm evaluation itself as an exploratory ensemble analysis problem. Rather than trusting one detector, the system generates many ensemble components, each corresponding to an anomaly detection algorithm with a particular parameter setting and sampled feature subspace.

The workflow begins with data preprocessing and feature bagging. The analyst loads a multidimensional dataset, and the system samples feature subspaces using methods such as random feature bagging, non-redundant feature bagging, or rotated bagging. These subspaces are important because different anomalies may only appear in certain combinations of dimensions. Next, several representative anomaly detection algorithms, including one-class SVM, k-nearest neighbors, local outlier factor, angle-based outlier detection, robust covariance estimation, and isolation forest, are run on the feature subspaces. Each resulting ensemble component produces an anomaly score ranking over the data instances.

The first analytical stage is a macro-level overview of the ensemble. EnsembleLens visualizes the overall similarity structure among ensemble components, enabling the analyst to see which detectors behave similarly and which ones are outliers. Components that fall far from the main cluster may be producing unusual rankings, either because they detect a genuinely different type of anomaly or because they perform poorly for the dataset. The analyst then moves to a meso-level correlation matrix, where pairs of ensemble components are compared more explicitly. Glyphs show rank correlation, shared top anomalies, and changes in the ranking of top-scored instances. This step supports reasoning about whether two algorithms agree on what is anomalous and whether disagreement is systematic.

At the micro level, the analyst inspects detailed ranked lists of anomalous instances from selected components. Here the workflow connects algorithm behavior back to the original data attributes: each ranked item is shown with its feature values, making it possible to judge whether top-ranked anomalies have unusual multidimensional profiles. The analyst can label suspicious or clearly normal instances, and this human feedback updates detector weights and the combined anomaly ranking. The system also provides detector and feature-subspace views so that the analyst can reason about whether performance differences arise from the algorithm type, parameter sensitivity, data characteristics, or selected features.

The workflow is iterative. Analysts repeatedly inspect ensemble structure, compare detector correlations, examine ranked instances, label examples, and update the ensemble combination. The goal is not only to produce a robust anomaly list, but also to understand which algorithms are appropriate for the dataset and why. EnsembleLens therefore turns unsupervised anomaly detector selection into a guided visual reasoning process.

27. explAiner workflow

explAiner: A Visual Analytics Framework for Interactive and Explainable Machine Learning
— T. Spinner et al., *IEEE Transactions on Visualization and Computer Graphics*, 26(1), 1064–1074, 2020. doi: 10.1109/TVCG.2019.2934629

The explAiner workflow is built around the question of how users can understand, diagnose, and refine machine learning models through an integrated visual analytics process. The paper argues that explainable AI methods are often available as isolated tools, but are not usually embedded into the practical model development workflow. explAiner

therefore proposes both a conceptual framework and a TensorBoard-based system in which explanation methods become part of an iterative machine learning pipeline.

The workflow begins with a model state, meaning a trained model with a particular architecture, parameters, and learned weights. Users first enter an understanding phase, where they inspect the model structure, inputs, outputs, and architectural components. This stage is especially important for model novices and users who need to understand what the model is composed of before interpreting its behavior. In the system implementation, this is supported by an interactive graph representation of the neural network and information cards that explain selected components. The purpose is to establish a mental model of the system being analyzed.

The next phase is diagnosis. Users apply one or more explainers, which are modular explanation components that take model states, model inputs, model outputs, or model internals as input and produce explanations. These explanations may be visualizations, feature attributions, saliency maps, surrogate models, line charts of internal activations, or other diagnostic views. Some explainers operate at high abstraction, explaining the whole model's behavior for an input example, such as LIME or layer-wise relevance propagation. Others operate at low abstraction, focusing on parts of the model such as tensors, weights, or intermediate layers. Through these explainers, users identify possible failures, biases, unstable behaviors, dead weights, unexpected feature reliance, or misleading predictions.

Once a problem or opportunity has been identified, the workflow moves into refinement. Refinement is conceptualized as a transition from one model state to another. A user may change architecture, parameters, training data, or other model properties, guided by explanations or system recommendations. The refined model becomes a new model state, which can again be understood and diagnosed. This produces an iterative loop: understand the model, diagnose its limitations, refine it, and repeat until the model is more suitable for the task.

The framework also includes global monitoring and steering mechanisms that run across the workflow. These include model quality monitoring, data shift scoring, search-space exploration, comparative analytics across model states, provenance tracking, reporting, and trust building. In the implemented system, provenance cards let users save explanation results, annotations, and refinement decisions, later assembling them into reports. The analytical logic is that explanation is not a one-time output but an iterative, inspectable, and documentable process. explAiner links XAI methods to interactive machine learning so users can move from seeing explanations, to forming diagnoses, to changing models in a traceable workflow.

28. Hybrid Visual Steering workflow

Visual Analytics for Complex Engineering Systems: Hybrid Visual Steering of Simulation Ensembles — K. Matković et al., IEEE Transactions on Visualization and Computer Graphics, 20(12), 1803–1812, 2014. doi: 10.1109/TVCG.2014.2346744

The Hybrid Visual Steering workflow addresses a core engineering problem: complex simulation models have large parameter spaces, and exhaustively sampling them is too expensive, yet engineers need to understand and optimize system behavior. The paper's

research question is how interactive visual steering can be combined with automatic regression-based optimization to guide simulation ensemble design more efficiently. The workflow is demonstrated with a common rail diesel injection system, but the analytical logic applies broadly to simulation-based design.

The workflow begins with an initial design of experiment. Engineers define a simulation model and identify control parameters that can vary, such as pipe length, rail volume, pressure, or injection timing. A first set of parameter combinations is sampled and simulated. Each simulation run produces complex outputs, often time-dependent curves. Because optimization methods and many overview visualizations require scalar values, the workflow extracts meaningful scalar features from these outputs, such as maximum pressure differences, injected fuel mass differences, or damping values. The resulting ensemble links each control-parameter vector to extracted output features.

The next step is interactive visual exploration of the ensemble. Engineers inspect the distributions of control parameters and output features using coordinated views, including the Parameters Exploration View and views for curves, scatterplots, and parallel coordinates. This stage supports sensitivity analysis, comparison of different parameter-space regions, and model reconstruction, where the engineer asks which parameters lead to desired output behavior. Brushing and linked views allow engineers to select parameter ranges or output patterns and immediately see corresponding effects elsewhere in the ensemble. This visual phase lets domain knowledge guide attention toward promising or problematic regions of the parameter space.

Once an interesting region is identified, the workflow builds a regression model as a surrogate for the simulation. The surrogate approximates simulation outputs continuously across a selected region, enabling faster optimization than running the full simulation repeatedly. Engineers then define objectives and constraints interactively, such as minimizing pressure differences or satisfying acceptable injection-curve shapes. An optimization component searches the regression model for apparently optimal parameter settings. However, because the surrogate is only an approximation, its result is treated as a hypothesis rather than a final answer.

The workflow then verifies and refines the optimization result. The proposed optimum is simulated with the original simulation model, and neighboring points may be sampled to test whether the surrogate prediction was reliable. The Regression Exploration View compares actual simulation results with regression predictions so engineers can judge model accuracy. If the optimum lies in a region where the regression model is inaccurate, the engineer selects additional sampling points, runs new simulations, updates the ensemble, rebuilds the regression model, and repeats the process.

Thus the workflow alternates between visual exploration, surrogate modeling, automatic optimization, and simulation-based validation. Its purpose is not to replace the engineer with automation, but to combine human intuition, visual pattern recognition, and computational optimization into a more efficient design loop.

29. Image-Based Projection Labeling for Mobile Eye Tracking

Image-Based Projection Labeling for Mobile Eye Tracking — K. Kurzahls, ACM Symposium on Eye Tracking Research and Applications (ETRA '21), Article 4, 2021. doi: 10.1145/3448017.3457382

The Image-Based Projection Labeling workflow addresses the problem that mobile eye tracking data is difficult and time-consuming to annotate with areas of interest. In screen-based studies with static stimuli, AOIs can be drawn once and reused across participants, but in mobile eye tracking every participant records a different egocentric video, with objects appearing, disappearing, and changing position. The paper's research question is how analysts can efficiently assign AOI labels to mobile gaze data and compare participants during and after annotation. The workflow replaces frame-by-frame boundary annotation with a visual process based on gaze-centered image thumbnails.

The workflow begins with recorded world-view videos and gaze coordinates from mobile eye tracking glasses. Because gaze sampling and video frame rates differ, the data is reduced to one gaze coordinate per video frame when possible. Around each gaze point, the system extracts a gaze thumbnail that captures the image content near the point of regard. Each thumbnail is then represented by a feature vector extracted from a convolutional neural network. This transforms raw gaze-over-video data into a set of image descriptors that can be compared computationally.

The next stage is interactive segmentation. Consecutive gaze thumbnails are compared by feature similarity to determine when the viewed content changes. If adjacent thumbnails remain similar, they are grouped into a segment, representing a likely dwell on the same AOI. If similarity drops below a threshold, a new segment is created. Rather than hiding this automatic step, the workflow visualizes it with local and global slitscans. Local slitscans show image content at gaze positions over time, while global slitscans summarize the world-camera video. Similarity plots can be overlaid, allowing analysts to adjust segmentation thresholds and understand how parameter changes affect the resulting segments.

After segmentation, each segment is represented by a thumbnail and projected into a two-dimensional space using dimensionality reduction, specifically UMAP in the implementation. The analytical assumption is that thumbnails showing the same AOI will have similar visual features and will appear close together in the projection. Analysts inspect this projection, identify clusters or neighborhoods of similar thumbnails, and select groups of segments for annotation. Because thumbnails can overlap in the projection, selected items can be rearranged into a grid for easier inspection and cleaning. The analyst then assigns an AOI label, such as a person, magazine cover, dashboard, or object category.

The labels are immediately propagated back to timeline views, where scarf plots show each recording's scanpath as a sequence of AOI labels. This gives continuous feedback on annotation progress and allows analysts to compare participants while labeling. The workflow is therefore iterative: segment videos, inspect slitscans, adjust thresholds, project thumbnails, label visually similar groups, validate labels in timelines, and refine uncertain cases. Its analytical logic is to combine machine-based visual similarity with human

semantic interpretation, reducing annotation effort while still allowing flexible AOI definition in unconstrained mobile settings.

30. IRVINE Workflow

IRVINE: A Design Study on Analyzing Correlation Patterns of Electrical Engines — J. Eirich et al., *IEEE Transactions on Visualization and Computer Graphics*, 28(1), 11–21, 2022. doi: 10.1109/TVCG.2021.3114797

The IRVINE workflow supports automotive engineers in detecting and understanding previously unknown manufacturing errors in electrical engines from acoustic test data. The paper’s research question is how visual analytics can help engineers analyze large numbers of produced engines, identify similar acoustic error signatures, label faulty engines, and preserve expert knowledge about error causes. The workflow begins with acoustic measurements collected during engine testing. Each engine is run through a controlled speed ramp, and sound measurements are transformed into spectrograms that represent acoustic behavior across engine speed and order. Orders correspond to mechanical subcomponents, such as gears, rotor shaft, or bearings.

The first analytical transformation is the computation of acoustic signatures. Instead of analyzing raw spectrograms alone, IRVINE derives residual measurements by comparing each engine to the ensemble distribution. It then computes correlations between selected order pairs associated with engine subcomponents. These correlations are aggregated into a so-called Hypermatrix, where cells represent relationships between subcomponent-related orders. The Hypermatrix acts as a compact signature of how acoustic anomalies in one component resonate or correlate with other components. This supports the domain question of distinguishing primary error symptoms from secondary symptoms.

The next workflow stage groups engines with similar signatures. Features are extracted from regions of the Hypermatrix, and a self-organizing map clusters engines by signature similarity. Engineers begin with a cluster overview, where each cell represents a group of engines with similar Hypermatrices. The overview also shows how many engines are in each cluster, how many have already been labeled, and how anomalous the cluster appears. This supports exploration when engineers do not know which engines to inspect, as well as confirmation when they search for known types of errors.

After selecting a cluster, engineers drill down to individual engines. An engine list shows each engine’s Hypermatrix and anomaly indicators, helping the engineer choose a specific engine for detailed analysis. The selected engine is then inspected through linked views: a detailed Hypermatrix, the original spectrogram, line charts, scatterplots, and bar charts. By hovering over cells in the Hypermatrix, the engineer maps correlation patterns back to raw acoustic orders in the spectrogram. This is the critical reasoning step: the engineer forms a hypothesis about an error source, such as a B-bearing defect, and validates it against the raw acoustic measurement.

Once an error is understood, the engineer assigns a categorical label to the engine and optionally annotates the exact region in the spectrogram that explains the label. Labels and annotations are stored in a knowledge database. This knowledge can guide later analyses, support label suggestions, and help other stakeholders understand why an

engine was classified as faulty. The workflow can then loop back: engineers can retrain the clustering on selected clusters or labels to obtain more fine-grained groupings. IRVINE's analytical logic is therefore to move from mass acoustic data, to correlated signatures, to clustered groups, to individual diagnosis, and finally to reusable expert knowledge.

31. ISeeCube: Visual Analysis of Gaze Data for Video

ISeeCube: Visual Analysis of Gaze Data for Video — Kuno Kurzhals, Florian Heimerl, and Daniel Weiskopf, *Proceedings of the Symposium on Eye Tracking Research and Applications (ETRA '14)*, 43–50, ACM, 2014. doi: 10.1145/2578153.2578158

The ISeeCube workflow addresses the challenge of analyzing eye tracking data recorded from dynamic stimuli such as video. The paper's research question is how analysts can understand where multiple viewers looked over time, identify attentional synchrony, define and analyze dynamic AOIs, and compare viewers' scanpaths. The workflow combines spatiotemporal visualization, AOI annotation, timeline-based inspection, and automated scanpath clustering.

The analysis can begin without predefined AOIs. Gaze data from multiple viewers is loaded together with the video stimulus, with gaze coordinates associated with video frames. ISeeCube first provides a space-time cube representation in which the two spatial dimensions of the video are combined with time as the third dimension. Gaze points, scanpaths, clusters, and later AOIs can be visualized in this cube. This gives analysts a static overview of an otherwise dynamic viewing process. Rather than watching the entire video sequentially, analysts can inspect where gaze points concentrate in space-time and identify intervals that may contain important viewing behavior.

The first major analytical task is finding attentional synchrony. ISeeCube uses density-based filtering of gaze points to highlight regions where multiple viewers looked at the same location at the same time. Such dense regions often correspond to salient moving objects or important events in the video. The workflow also applies spatiotemporal clustering to gaze points, helping identify potential AOIs in scenes where attention is distributed among multiple objects. These clusters serve as visual cues for where meaningful dynamic AOIs may exist.

The next step is AOI annotation. Using an integrated editor, the analyst defines dynamic axis-aligned bounding boxes over video objects, with key positions interpolated over time. AOIs can be informed by the previously detected gaze clusters, meaning that observed viewer attention helps guide annotation. Once AOIs are defined, they can be represented both in the space-time cube and in an AOI timeline. The timeline shows when each AOI exists and overlays attention histograms indicating how many viewers looked at it over time. This transforms raw gaze coordinates into semantically meaningful object-based attention data.

The workflow then supports viewer-level comparison. Each viewer is represented by a scarf plot showing which AOI they looked at frame by frame. Analysts select time spans and compare scanpaths across viewers. ISeeCube computes similarity between scanpaths using different measures: Levenshtein distance for AOI order and temporal alignment, attention-map similarity for overall distribution of attention, and transition-map similarity for

movement between AOs. Hierarchical clustering groups viewers according to these measures, and the resulting dendrogram is linked to the scarf plots.

The analytical process is iterative. Analysts identify time spans of synchrony or distributed attention, annotate or refine AOs, inspect timelines, compare viewers, and detect outliers or viewer groups. ISeeCube therefore bridges unlabeled gaze exploration and AOI-based scanpath analysis, allowing researchers to answer what viewers looked at, when they looked at it, and how their viewing strategies differed.

32. IVESA – Visual Analysis of Time-Stamped Event Sequences

IVESA – Visual Analysis of Time-Stamped Event Sequences — J. Bernard et al., IEEE Transactions on Visualization and Computer Graphics, 31(4), 2235–2256, 2025. doi: 10.1109/TVCG.2024.3382760

The IVESA workflow addresses a specific kind of temporal data: time-stamped event sequences that contain event times but no event values. The paper’s research question is how domain experts can visually analyze large collections of such sequences, discover temporal occurrence patterns, relate them to contextual metadata, and reduce data complexity through metrics, filtering, clustering, and motif simplification. The workflow is designed to be general across domains such as photography, computer networks, permafrost observations, radiology, stock-market tweets, and wastewater treatment.

The workflow begins with data preparation. Long event streams may be segmented into meaningful sequences, such as days, months, years, devices, sensors, or workstations, depending on the domain. Sequences are then aligned either in absolute time, relative time, or another meaningful temporal frame so that they can be compared. IVESA also computes metrics that describe event behavior without relying on event values. Some metrics operate at the event level, such as velocity, acceleration, density, entropy, or outlieriness around each timestamp. Others summarize entire sequences, producing features such as regularity, duration, event count, or aggregate temporal behavior.

The first interactive stage is overview. IVESA shows all sequences in a scalable sequence overview, where each event is represented as a small temporal mark along a shared time axis. Users sort or rank sequences by metadata, features, clusters, or other criteria. This supports rapid discovery of patterns such as gaps, bursts, periodic behavior, seasonality, or anomalous sequences. The analyst can select interesting sequences and inspect them in a details view, where individual events, temporal features, and motifs can be examined more closely. This implements the classic logic of overview first, then details on demand.

A major part of the workflow is contextualization through metadata. IVESA provides a metadata view based on heatmaps and small multiples. Users map metadata or features to axes, bin values, and compare event distributions across categories, time periods, devices, locations, or clusters. This helps explain why an event pattern occurs. For example, network workstation activity may differ by device type, or permafrost melting events may vary by borehole depth and season. Thus, the workflow connects temporal event patterns to external explanatory attributes.

The system then supports data simplification. Users can filter sequences by metadata or features to focus on relevant subsets. They can cluster sequences based on selected metrics, inspect the resulting hierarchy, adjust aggregation level, and propagate cluster labels back to other views. They can also define motifs by selecting subsequences of events in the details view, choose similarity and window parameters, and let IVESA find matching motifs across the dataset. Repeated motifs can then be visually substituted, reducing clutter and emphasizing higher-level temporal structure.

The workflow is iterative and exploratory. Analysts move between overview, sorting, selection, detail inspection, metadata contextualization, filtering, clustering, and motif definition. Its analytical logic is that value-free event data must be understood through timing, recurrence, density, gaps, and contextual metadata. IVESA provides the metrics and visual interfaces needed to transform raw timestamp collections into interpretable temporal patterns and domain insights.

33. Jigsaw: Supporting Investigative Analysis through Interactive Visualization

**J. Stasko, C. Görg, and Z. Liu. *Information Visualization*, vol. 7, no. 2, pp. 118–132, 2008.
doi: 10.1057/palgrave.ivs.9500180**

The workflow described in the Jigsaw paper addresses investigative sensemaking over collections of short text documents, such as intelligence reports, news articles, or case files. The paper’s central research question is how interactive visualization can help analysts discover connections among people, places, organizations, dates, and events that are scattered across many documents. The workflow therefore begins not by replacing reading, but by transforming a document collection into a visual index that helps analysts decide what to inspect next and how separate pieces of evidence might fit together.

The first analytical step is to import a document collection and extract named entities from each document. Entities such as persons, locations, organizations, dates, and other domain-specific categories become the main units of analysis. The system assumes that if two entities appear in the same document, they may be meaningfully connected. This co-occurrence model is deliberately simple, but it gives the analyst a starting structure for exploration. Because automatic entity extraction is imperfect, the workflow includes correction: analysts can inspect original documents, add missing entities, remove false ones, and change entity types. This ensures that later visual links correspond more closely to meaningful evidence.

Once entities have been extracted, the analyst usually begins with a seed: a known person, organization, location, date, or keyword of interest. Jigsaw then supports a foraging loop in which the analyst searches for this seed, views related entities, identifies potentially important documents, reads those documents, and expands from newly discovered entities. The List View helps compare entity types side by side, for example showing which people are connected to a suspect and which organizations are connected to those people. Stronger co-occurrences are visually emphasized, allowing the analyst to prioritize relationships that recur across documents. The Graph View supports a more associative

exploration: the analyst expands an entity to reveal documents mentioning it, then expands those documents to reveal additional entities, gradually following chains of evidence.

Other views support alternative analytical angles. The Document View grounds the process in the original text, allowing analysts to verify whether a visual connection is meaningful. The Calendar View places documents and entities in temporal context, helping analysts notice patterns that are not captured by document-level co-occurrence alone. The Scatter Plot View highlights pairwise relationships between entity types, while the Document Cluster View helps organize or filter the document set. As analysts identify relevant evidence, they can place entities, documents, and hypotheses into the Shoebox, which supports evidence marshalling and hypothesis construction.

The workflow is iterative and human-centered. Jigsaw does not automatically infer a final plot or conclusion; instead, it helps analysts move between overview, connection discovery, document reading, temporal reasoning, and evidence organization. This logic directly supports the paper's aim: to reduce the cognitive burden of investigative analysis by externalizing entity relationships, guiding document selection, and helping analysts build and refine hypotheses from distributed textual evidence.

34. Kokiri: Random-Forest-Based Comparison and Characterization of Cohorts

K. Eckelt, P. Adelberger, M. J. Bauer, T. Zichner, and M. Streit. *IEEE VIS Workshop on Visualization in Biomedical AI*, 2022. doi: 10.1101/2022.08.16.503622

Kokiri presents an analytical workflow for comparing and characterizing patient cohorts using high-dimensional biomedical data, especially genomic mutation data. The paper's motivating research question is how analysts can understand why patient subgroups differ when many attributes may interact, and when simple one-attribute-at-a-time comparison may miss important combinatorial effects. The workflow therefore uses a random forest classifier not primarily as a predictive tool, but as an analytical instrument for exposing which variables distinguish cohorts, how separable the cohorts are, and which patients are typical or atypical members of their groups.

The workflow begins outside Kokiri with the definition of cohorts. In the paper's use case, cohorts are lung cancer patients grouped by race and gender, but conceptually the workflow applies to any tabular dataset with user-defined subgroups. Once cohorts are selected, the first step inside Kokiri is to assess overlap between them. This matters because overlapping cohorts complicate interpretation: if the same patient belongs to multiple groups, then perfect separation may be impossible or analytically misleading. The Overlap View establishes whether the comparison is between distinct cohorts, partially overlapping cohorts, or subset relationships.

The next step is cohort comparison. Kokiri trains a random forest classifier to distinguish the selected cohorts based on the chosen attributes, such as mutation status or metadata. If the classifier can separate the cohorts well, this suggests that the data contain systematic differences between them. If it performs poorly, the cohorts may be similar with respect to

the selected attributes, or the attributes may not capture the relevant distinctions. Kokiri ranks attributes by their importance in the classifier, allowing analysts to see which genes or variables contribute most to separating the cohorts. Importantly, this ranking reflects multivariate decision structure rather than isolated pairwise statistics. The analyst can inspect attribute distributions across cohorts to understand why the model considers a feature important.

The workflow then shifts from comparing cohorts to characterizing them. Kokiri examines the random forest's prediction probabilities for individual patients. Patients that the classifier assigns confidently to their true cohort can be interpreted as typical examples under the selected data representation. Patients assigned with low confidence, or assigned more strongly to another cohort, become analytically interesting: they may be outliers, members of sub-cohorts, or evidence that cohort boundaries are not clean. A scatterplot embedding of prediction probabilities gives an overview of cohort homogeneity, overlap, and unusual cases, while an item ranking helps locate difficult-to-classify patients.

The process is explicitly iterative. After seeing important attributes, separability, and outliers, users can refine the analysis by excluding known differences, changing the attribute set, or redefining cohorts. In the lung cancer case, this workflow confirmed known differences such as EGFR and KRAS mutation patterns, but also revealed differences caused by missing sequencing data, such as FGFR4 and BRCA1 coverage differences. Thus Kokiri's analytical logic is not merely to classify patients, but to use classification behavior to guide biomedical interpretation, quality assessment, and further cohort refinement.

35. LFPeers: Temporal Similarity Search and Result Exploration

M. Sachdeva, J. Burmeister, J. Kohlhammer, and J. Bernard. *Computers & Graphics*, vol. 115, pp. 81–95, Oct. 2023. doi: 10.1016/j.cag.2023.06.009

LFPeers describes a workflow for learning from temporal peers: identifying objects that behaved similarly up to a chosen point in time and then studying how and why they diverged afterward. The paper applies this logic to Covid-19 country data, but the analytical structure is domain-general. Its central research question is how visual analytics can support cause-and-effect reasoning in temporal, multimodal data, especially when analysts want to compare countries that were initially similar but later experienced different outcomes.

The workflow begins with the selection of a focal object, such as a country whose pandemic trajectory the analyst wants to understand. This selected object becomes the reference for similarity search. The analyst then defines a search period ending at a meaningful time point, called t_0 . This time point separates the workflow into two phases. Before t_0 , the goal is similarity: find other countries that resembled the reference country according to selected temporal attributes. After t_0 , the goal changes to explanation: analyze how those similar countries subsequently diverged and what external factors may account for the differences.

During the search phase, the analyst chooses one or more temporal attributes, such as confirmed cases, policy stringency, mobility, or other Covid-related indicators. For each

attribute, the analyst defines a time range and a similarity model. LFPeers then computes nearest neighbors to the reference country over that pre- t_0 interval. The analyst can inspect the ranked list of similar countries, view their trajectories against the global context, and refine the peer group. Similarity is not treated as a fixed algorithmic result; the analyst can adjust descriptors, distance metrics, and criteria. Multiple similarity criteria can be combined, so a peer group might consist only of countries that were similar in infection trajectory, government stringency, and mobility behavior before t_0 .

Once a peer group is established, the workflow moves to the exploration phase. Here the analyst investigates what happened after t_0 . Instead of looking for similarity, the analyst now looks for variation: which peers continued along the same trajectory, which diverged, and when. Bundled time-series views reveal “fan-out” behavior, where initially similar countries begin to separate. This shift from similarity search to divergence analysis is the core analytical move in the paper. It allows analysts to ask not simply “which countries are similar?” but “what changed after they were similar?”

The final step is to relate post- t_0 divergences to possible explanatory factors. LFPeers supports correlation analysis at selected later time points, helping analysts compare outcome differences with external attributes such as policies, mobility changes, or contextual variables. This does not prove causation, but it supports hypothesis generation about which factors may explain why peers diverged. In the Covid-19 scenario, an analyst might select the United States, find countries with similar early case trajectories, refine the peer set by policy and mobility similarity, and then examine which later policy or behavioral differences correlate with worse or better outcomes. The workflow thus operationalizes “learning from peers” as a sequence of reference selection, pre-event similarity search, peer refinement, post-event divergence exploration, and explanatory correlation.

36. MotionExplorer: Exploratory Search in Human Motion Capture Data Based on Hierarchical Aggregation

J. Bernard, N. Wilhelm, B. Krüger, T. May, T. Schreck, and J. Kohlhammer. *IEEE Transactions on Visualization and Computer Graphics*, vol. 19, no. 12, pp. 2257–2266, Dec. 2013. doi: 10.1109/TVCG.2013.178

MotionExplorer presents an analytical workflow for exploring large human motion capture collections, with the goal of helping motion researchers find reusable motion sequences and understand variations in motion style. The paper’s research problem is that motion databases are too large to inspect manually, while existing retrieval methods often fail to provide an intuitive overview of available motion types and style variations. The workflow therefore combines hierarchical aggregation, visual exploration, and query-by-example search.

The workflow begins by treating motion capture data as multivariate time series. Each motion sequence is decomposed into individual human poses, and each pose is represented by a feature vector derived from marker positions. This transforms complex motion recordings into a searchable pose space. The system then clusters poses hierarchically, producing aggregates of similar poses at different levels of detail. This

aggregation is essential because individual databases may contain tens of thousands of poses; analysts need a global overview before they can identify useful subsets.

The first major analytical step is overview exploration. The pose hierarchy is shown as a dendrogram, where analysts can adjust the aggregation level. At coarse levels, they see broad categories of pose states; at finer levels, they can inspect more detailed pose variations. This supports the question “What kinds of poses exist in the collection?” The analyst can also filter out irrelevant clusters and focus on regions of the pose hierarchy that correspond to motions of interest.

The next step is to understand how poses connect over time. MotionExplorer constructs a motion graph in which pose clusters are nodes and observed transitions between poses are edges. This shifts the analysis from static pose similarity to dynamic motion structure. Analysts can see which pose states are connected by actual motion sequences, identify cycles, detect missing transitions, and assess whether the database contains the necessary building blocks for motion synthesis. This directly supports the practical research question of whether a desired motion can be reused from existing recordings or whether new data must be captured.

After exploring the overview, the analyst formulates a visual query. Instead of typing metadata, the user selects example start and end poses from the visualization. The system retrieves motion subsequences that begin and end with matching pose clusters. This query-by-example workflow is important because motion researchers often think in terms of body configurations and movement transitions rather than textual labels. The search results are then displayed as temporal sequences, allowing analysts to compare multiple retrieved motions.

The final analytical step is style-variation exploration. Retrieved sequences may all satisfy the same start-end query, but differ in speed, dynamics, intermediate poses, or execution style. MotionExplorer supports inspection of these differences through aligned temporal views, animation, and filtering. Analysts can remove irrelevant results, focus on promising sequences, and compare how alternative paths realize the same motion goal. In the field study, users employed this workflow to rediscover known motions, identify unexpected style variations, find missing recordings, and locate connection poses for motion synthesis. Thus the workflow moves from pose-level abstraction, to motion-transition overview, to visual querying, to detailed comparison of motion style, matching the needs of researchers who must reuse and synthesize human motion from large capture repositories.

37. A Multimodal Framework for Understanding Collaborative Design Processes

Maurice Koch, Nelusa Pathmanathan, Daniel Weiskopf, and Kuno Kurzhals. *IEEE Transactions on Visualization and Computer Graphics*, vol. 32, no. 1, pp. 582–592, 2026. doi: 10.1109/TVCG.2025.3634232

This paper describes a workflow for analyzing collaborative design workshops using multimodal data. Its central research question is how researchers can capture, summarize,

and communicate important findings from a collaborative design process, including how those findings emerged through participant interaction. The workflow addresses a limitation of traditional workshop analysis: final notes, summaries, or photographs often record outcomes but not the temporal process by which participants negotiated, discussed, and produced them.

The workflow begins with workshop planning and data acquisition. The authors recommend instrumenting a collaborative working area, such as a table covered with paper, sketches, and materials. Static areas of interest are defined in the workspace so that later activity and attention can be associated with meaningful regions. Depending on the workshop goals and practical constraints, the setup can include video, audio, eye tracking, and digital notes. Video captures physical activity such as pointing, drawing, and manipulating materials. Audio captures discussion and is transcribed. Eye tracking captures visual attention, including shared attention. Notes provide the subjective observations of workshop organizers.

After the workshop, the workflow moves to artifact extraction. The system derives structured artifacts from the raw multimodal recordings. From video, it detects hand activity in areas of interest. From gaze, it measures visual attention to those same areas. From audio, it produces transcripts and speaker-role information. From notes, it extracts time-stamped observations. A key step is topic segmentation: the long workshop recording is divided into meaningful temporal segments. Initial segmentation is based on changes in activity or attention, and refined using topic shifts in the transcript. This creates manageable units for later analysis.

The next stage is visual analysis in the reCAPit system. Analysts first obtain a topic overview through topic cards and timeline-based representations. Topic cards summarize automatically detected segments, while streamgraphs show how activity and attention move across regions of the workspace over time. This helps analysts identify phases such as brainstorming, explanation, consolidation, or decision-making. The timeline aligns utterances, gaze, activity, notes, and topic segments, allowing the analyst to move between high-level summaries and raw evidence.

The workflow then becomes interpretive and authorial. Analysts inspect relevant segments, filter topic cards by keywords, and drill down into raw video, transcript, gaze, or notes. They can enrich topic cards by adding quotes, screenshots, heatmaps, and their own interpretive notes. This turns topic cards into evidence-backed summaries of how a particular outcome emerged. In the band-practice case study, for example, the analyst traced how “segmentation” became a design requirement by following related discussions, sketches, attention patterns, and final consolidation notes. In the urban-planning case, the workflow helped connect verbal references, gestures, and visual activity around the concept of co-production.

The final step is communication. Authored topic cards and multimodal summaries provide a transparent account of the design process, not just its results. The analytical logic is therefore both bottom-up and top-down: automatic extraction and visualization reduce the burden of navigating large recordings, while human analysts retain responsibility for

interpretation, selection, and reporting. The workflow supports more rigorous design-study reporting by linking workshop outcomes to the multimodal evidence from which they arose.

38. Visual Analysis of Multivariate State Transition Graphs

A. J. Pretorius and J. J. van Wijk. *IEEE Transactions on Visualization and Computer Graphics*, vol. 12, no. 5, pp. 685–692, Sept./Oct. 2006.

This paper presents a workflow for analyzing large state transition graphs whose nodes have multiple attributes. The research question is how users can understand the behavior of complex transition systems, such as industrial controllers, when the graph contains tens of thousands of states and hundreds of thousands of transitions. Rather than relying only on graph topology, the workflow uses state attributes as the primary analytical handle, allowing users to explore the system from semantically meaningful perspectives.

The workflow begins with a state transition graph in which each node represents a system state and each directed edge represents a transition. Each state has attributes, such as the contents of machine components, variable values, or derived graph properties like fan-out. The analyst first selects a subset of attributes and orders them according to the perspective they want to take on the system. This selection is analytically important: choosing attributes related to a tray, a processing stage, robot arms, or deadlock status changes the way the system is partitioned and therefore the behavior that becomes visible.

The next step is interactive hierarchical clustering. The complete set of states forms the root of a hierarchy. The first selected attribute partitions the states by its values; the second attribute subdivides each partition; and so on. This creates a hierarchy of state clusters that is meaningful because it corresponds to domain-relevant attribute values. The analyst can also cluster attribute values themselves to simplify the analysis, for example grouping all nonzero fan-out values together and separating them from deadlock states.

The visualization then combines three linked views of the clustering result. The hierarchy shows how states have been partitioned by attributes. A bar tree shows quantitative metrics, especially the number of states in each cluster, so users can see how large or small different behavioral regions are. An arc diagram shows transitions between leaf clusters, with arc direction and thickness indicating the structure and frequency of transitions. The key analytical logic is that hierarchy, metric size, and transition behavior are viewed together, allowing users to relate what states are, how many there are, and how the system moves among them.

The workflow is iterative and perspective-driven. In the wafer-stepper case, the analyst first views the system from the tray's perspective, revealing deterministic forward progress from unprocessed to processed wafers. Then the analyst views it from the wafer-stage perspective to verify requirements, such as whether unprepared wafers can reach the processing stage. Then the locks' perspective reveals symmetry and repeated behavioral patterns. Each perspective answers a different behavioral question by changing the selected attributes and reading the resulting hierarchy and arcs.

For focused analysis, the workflow supports anomaly investigation, especially deadlock detection. The analyst clusters first by fan-out, isolating states with no outgoing transitions. Then additional attributes are added to identify the conditions under which deadlock occurs. By zooming into the deadlock cluster, inspecting incoming arcs, and tracing their source clusters, the analyst reconstructs the exact preconditions leading to deadlock. Thus the workflow moves from semantic attribute selection, to hierarchical abstraction, to transition-pattern inspection, to focused causal analysis of problematic states. It helps users understand large transition systems by making behavioral structure visible through attribute-based abstraction.

39. NodeTrix: A Hybrid Visualization of Social Networks

N. Henry, J.-D. Fekete, and M. J. McGuffin. *IEEE Transactions on Visualization and Computer Graphics*, vol. 13, no. 6, pp. 1302–1309, Nov.–Dec. 2007. doi: 10.1109/TVCG.2007.70582

NodeTrix proposes an analytical workflow for exploring social networks that are globally sparse but locally dense. The paper’s research question is how to visualize networks so that analysts can understand both the global structure of communities and the detailed internal relationships within those communities. Traditional node-link diagrams show global structure well but become unreadable in dense clusters; adjacency matrices show dense communities well but make paths and global structure difficult to follow. NodeTrix combines both representations in one workflow.

The workflow typically begins with a social network shown as a node-link diagram. This gives analysts an initial overview of the network’s global structure: which actors or communities are connected, which parts of the network are peripheral, and which nodes appear to bridge groups. The analyst then identifies candidate communities, either visually or through interaction, and aggregates selected nodes into matrix representations. This is a key analytical step: dense subgraphs are transformed into adjacency matrices, while relationships between communities remain visible as links. The result is a hybrid view in which local density is handled by matrices and global connectivity remains node-link based.

Once communities are represented as matrices, the analyst can examine internal structure more clearly. Matrix cells show relationships among actors inside the community, making cliques, missing ties, dense blocks, and collaboration patterns easier to detect than in a tangled node-link cluster. At the same time, links from matrix edges to other nodes or matrices show which specific actors connect the community to the rest of the network. This supports the main social network tasks described in the paper: identifying communities, identifying central actors, and interpreting roles and positions.

The workflow is highly interactive. Analysts can lasso nodes to aggregate them into a matrix, drag a node into an existing matrix, extract a node from a matrix, move a node from one community to another, merge matrices, or reorder matrix rows and columns. These operations allow community boundaries to be tested rather than assumed. For example, if removing one actor disconnects two dense subgroups, that actor may function

as a bridge. If moving an actor into a matrix reveals many internal ties, the actor may belong analytically to that community. The workflow therefore treats visualization as a way to manipulate and test social interpretations.

NodeTriX also supports presentation after exploration. Analysts can adjust matrix size, labels, link thickness, and visual encodings to communicate findings. In the InfoVis co-authorship case study, the workflow reveals different collaboration patterns. Some communities show a “cross” pattern, where one central researcher collaborates with many others who do not collaborate much with each other. Other communities show dense blocks, indicating a group of researchers who all collaborate strongly. Intermediate cases reveal both central figures and cohesive subgroups.

The analytical logic is thus to begin with global structure, convert dense local regions into readable matrices, interactively refine community membership, inspect internal and external ties, and interpret collaboration roles. NodeTriX directly addresses the paper’s research problem by preserving the strengths of both node-link and matrix views in a single exploratory workflow.

40. ParaGlide: Interactive Parameter Space Partitioning for Computer Simulations

S. Bergner, M. Sedlmair, T. Möller, S. N. Abdolyousefi, and A. Saad. *IEEE Transactions on Visualization and Computer Graphics*, vol. 19, no. 9, pp. 1499–1512, Sept. 2013. doi: 10.1109/TVCG.2013.61

ParaGlide describes a workflow for exploring parameter spaces of computational simulation models. The paper’s central research question is how model developers can understand which regions of a high-dimensional input parameter space produce qualitatively different model behavior. Unlike workflows focused only on finding a single optimal parameter setting, ParaGlide emphasizes parameter space partitioning: discovering regions of input settings that correspond to distinct output patterns.

The workflow begins by connecting ParaGlide to an existing simulation model. The model has input parameters, a simulator that produces outputs, and optionally derived variables or features computed from those outputs. This integration is important because model developers often work in environments such as MATLAB or R; ParaGlide is designed to let them continue using their models while gaining an interactive visual interface for exploration.

The next step is data generation. Since simulation data may not already exist, the analyst defines a region of interest in the input parameter space. ParaGlide supports region-based sampling: the user selects parameter ranges, chooses a sampling strategy, and runs the simulator on generated parameter configurations. This creates experimental points, each consisting of input settings and corresponding model outputs. The workflow is iterative because simulations can be expensive. Analysts begin with coarse sampling, inspect results, then refine or resample promising regions rather than uniformly covering the entire parameter space.

After outputs are generated, the analyst inspects them through linked views such as histograms, scatterplot matrices, and detail views of individual outputs. This supports basic model analysis: seeing which regions have been sampled, which outputs appear interesting, and how parameters relate to derived variables. In the biological modeling case, for example, users could inspect simulated animal aggregation patterns and relate them to parameter settings. In the image segmentation case, users inspected segmentation outputs and performance measures. In the fuel cell case, users examined plots describing cell behavior.

The core analytical step is output-based clustering. ParaGlide helps users compare high-dimensional or complex outputs by computing derived features, distances, and low-dimensional embeddings. The analyst then manually identifies groups of similar outputs in the embedding. These groups are not merely clusters in output space; when brushed back onto the input parameter views, they reveal regions of the input space that produce similar behavior. This is the parameter space partitioning step. It transforms the question “What does this parameter setting do?” into “Which regions of parameter space produce this type of behavior?”

The workflow then supports interpretation and refinement. Analysts examine the shape, size, and boundaries of the input-space partitions. Large regions suggest robust behavior; narrow regions suggest sensitivity to parameter changes. If a good output cluster occupies a stable region, it may be a better candidate than an isolated optimum. If clusters separate clearly by certain parameters, those parameters are influential; if a parameter varies widely within a good cluster, it may be less important. Users can then adjust the model, refine sampling, add derived variables, or compare model versions.

Across the biological modeling, image segmentation, and fuel cell case studies, the workflow helped users move from trial-and-error parameter tuning toward systematic exploration. Its analytical logic is to generate data selectively, view outputs in context, group similar behaviors, map those groups back to input space, and use the resulting partitions to reason about sensitivity, robustness, optimization, and model understanding.

41. Progressive and Interactive High-Dimensional Data Analysis

Designing Progressive and Interactive Analytics Processes for High-Dimensional Data Analysis — C. Turkay, E. Kaya, S. Balcisoy, and H. Hauser. *IEEE Transactions on Visualization and Computer Graphics*, 23(1), 131–140, 2017. doi: 10.1109/TVCG.2016.2598470

This paper describes an analytical workflow for high-dimensional data analysis in which the central problem is not only how to compute projections or clusters, but how to keep the analyst continuously engaged while those computations are still incomplete. The workflow begins with a high-dimensional dataset and an exploratory question, such as finding meaningful customer or transaction segments in financial data. Rather than requiring the analyst to choose parameters, wait for a full clustering or dimensionality reduction result, inspect it, and then restart the process, the paper proposes a progressive workflow. The analyst initiates an operation, for example by selecting a subset of variables,

choosing a region in a projection, or requesting a clustering. The system immediately computes an approximate result from a sample of the data and displays it within roughly one second.

The next step is interpretation under uncertainty. The analyst inspects the early projection or clustering result while the system continues to process more data in the background. Visualizations update progressively, often through animated transitions, so the analyst can see whether structures stabilize, disappear, or change substantially. This supports a key analytical decision: if the early result is unpromising, the analyst can abandon it quickly and try another feature selection or parameter configuration; if it appears promising, the analyst can wait for refinement or use the intermediate result to continue exploring. The research question is therefore addressed through rapid hypothesis generation and rejection, rather than through isolated, fully completed computational runs.

In the financial case study, this logic is instantiated as a segmentation workflow. Analysts begin by selecting variables such as demographic or financial features. Progressive clustering automatically proposes subsegments, while progressive PCA supports user-defined subsegments through visual selection of separated groups or outliers. These subsegments are then combined through Boolean operations into composite segments. Analysts further refine them with linked views, such as geographic maps or auxiliary plots, and then characterize the resulting segment using a difference view that compares selected transactions or customers against the rest of the dataset. The workflow cycles between feature choice, approximate computational response, visual inspection, subsegment selection, refinement, and interpretation.

The paper also introduces keyframed brushing as a more structured mode of exploration. Instead of manually dragging a selection and continuously monitoring linked views, the analyst defines start and end selections, and the system generates an animated sequence between them. This lets the analyst systematically observe how projections, statistics, or linked visual summaries change across a controlled range of selections.

The analytical logic of the paper is grounded in human time constraints. Immediate approximate feedback supports uninterrupted dialogue, animations support perceptual continuity, and bounded task sequences support sustained reasoning. The workflow's purpose is to make high-dimensional exploratory analysis faster and more productive by allowing analysts to act on provisional results, while still making progress and uncertainty visible. The authors also show that this approach can be misleading when unstable intermediate results are overinterpreted, so the workflow includes progress indicators, coarser visual encodings for uncertain states, and controls for pausing or managing progression.

42. QVis: Query-Based Visual Analysis of Multiscale Patterns in Spatiotemporal Ensembles

QVis: Query-Based Visual Analysis of Multiscale Patterns in Spatiotemporal Ensembles — R. Bauer, Q. Q. Ngo, G. Reina, S. Frey, and M. Sedlmair. IEEE Transactions on Visualization and Computer Graphics, 2025. doi: 10.1109/TVCG.2025.3629575

QVis describes a workflow for exploring large spatiotemporal ensembles by iteratively defining, retrieving, comparing, and interpreting dynamic patterns. The research problem is that scientific ensembles, such as videos of droplet impact experiments, contain patterns that vary in shape, size, location, and duration, and researchers need to understand how those patterns relate to experimental parameters. The workflow therefore begins before interactive analysis with a learned similarity model. The system trains a self-supervised model on spatiotemporal patches so that visually and dynamically similar patterns have nearby latent representations. Unlike earlier approaches, the model supports variable-sized patches, which is crucial because the relevant patterns may be small droplets, expanding crowns, splashes, bubbles, or longer events.

During analysis, the researcher first explores the ensemble visually. A direct data view shows an ensemble member at a selected time step, and the user inspects the data to identify a pattern of interest. The analyst then draws a bounding box around an example pattern and adds it as a positive query example. If needed, negative examples are added to exclude visually similar but analytically irrelevant patterns, such as background or confusing structures. The system compares the query examples against a precomputed search space of patches from the ensemble and returns ranked matches. The analyst adjusts a threshold to determine which retrieved patches count as occurrences of the queried pattern.

The next stage is pattern validation and refinement. The query interface displays example patches, ranked results, and score distributions, allowing the user to judge whether the learned metric is returning meaningful matches. If the results include false positives or miss important cases, the analyst adds more examples, adjusts thresholds, or defines additional queries. In the droplet impact case study, this process begins with obvious patterns such as drop, crown, crown-splash, and splash, and later expands to background, bubbles, jetting, and bubble-splash.

Once multiple queries are established, the workflow shifts from finding individual matches to understanding ensemble-wide pattern dynamics. The ensemble explorer shows, for each ensemble member, a timeline of when each queried pattern occurs. This lets the analyst compare temporal behavior across experiments, identify strong or weak occurrences, and detect members whose dynamics differ from the rest. The system also converts each member's timeline into a compact pattern sequence, such as drop → crown → splash → bubble-splash, which summarizes the temporal order of dominant patterns.

The guided exploration step helps ensure that the analyst has not overlooked important patterns. A clustering view groups sampled patches in the learned similarity space and uses glyphs and contours to show which clusters are already explained by existing queries. Uncolored or weakly covered clusters indicate underexplored regions. The analyst inspects these regions, discovers new patterns, and defines new queries, gradually improving coverage of the ensemble's dynamic behavior.

Finally, QVis connects pattern sequences to ensemble parameters. Members with similar pattern sequences are grouped, and their positions are shown in parameter space, such as Weber number and film thickness for droplet impact experiments. This enables the

researcher to build a characteristic map linking physical parameters to observed dynamic outcomes. The workflow thus moves from local pattern examples to ensemble-wide occurrence analysis and then to parameter-pattern interpretation, directly supporting the paper's question of how multiscale dynamic patterns vary across spatiotemporal ensembles.

43. RelEx: Visualization for Actively Changing Overlay Network Specifications

RelEx: Visualization for Actively Changing Overlay Network Specifications — M. Sedlmair, A. Frank, T. Munzner, and A. Butz. *IEEE Transactions on Visualization and Computer Graphics*, 18(12), 2729–2738, 2012. doi: 10.1109/TVCG.2012.255

RelEx describes an analytical workflow for automotive engineers who must change and optimize in-car communication network specifications. The research question is how visualization can support users who are not merely analyzing an existing network, but actively modifying an overlay communication network in which logical signal specifications must be mapped onto a physical network of electronic control units and bus systems. The workflow begins with a change request or optimization need, such as adding an electronic control unit, moving a signal, replacing a component, or reducing inefficient communication across bus systems.

The first analytical step is orientation in the physical network. Engineers use a topology view that resembles their familiar manually drawn network maps, showing electronic control units and bus connections. This supports the initial understanding of where components are located and how they are physically connected. However, the key analytical challenge is not the physical topology itself, which is relatively small, but the large number of logical signals and signal paths routed over it. The workflow therefore moves from physical orientation to exploration of communication relationships.

The central overview is a matrix view of the logical communication network. Sending electronic control units are placed along one axis and receiving units along the other; matrix cells show how many signals pass between each pair. Engineers inspect this view to find communication hotspots, local or global communicators, and broad communication patterns. For example, large cells indicate pairs of units that exchange many signals and may need attention during optimization. Sorting the matrix by bus system allows the analyst to see whether a bus is mostly internally communicating or heavily connected to other buses. This directly supports the traffic optimization question: which parts of the logical communication create expensive, overloaded, or unnecessary routing over the physical network?

After identifying a suspicious or important communication relationship, the engineer drills down. Selecting a matrix cell filters related information in coordinated list views, showing the specific signals exchanged between the two units. Selecting a signal then opens or updates a signal-path view that shows how that logical signal is routed through the physical network, including gateways and bus systems. This step is crucial because the

impact of a change request often lies not in the directly changed component but in the paths affected across the overlay network.

The workflow is iterative. Engineers select an electronic control unit, bus, signal, or communication relationship, inspect linked highlights across topology, matrix, list, and signal-path views, and then reason about the consequences of possible modifications. They may ask which communication partners would be affected if a unit were replaced, whether two units could be consolidated, which bus a new unit should connect to, or whether a signal could be rerouted to reduce gateway traffic. The visualization does not automatically optimize the network; instead, it externalizes complex relations that engineers previously had to reconstruct through repeated database queries and mental integration.

The analytical logic of RelEx is therefore relation-centered. It moves from overview of logical communication, to identification of significant patterns, to detailed tracing of signal paths, and finally to engineering judgment about network changes. This supports the paper's broader contribution: showing that network visualization workflows can focus on active change, routing, and path scalability, rather than only on cluster or centrality analysis common in social network visualization.

44. SOMFlow: Guided Exploratory Cluster Analysis

SOMFlow: Guided Exploratory Cluster Analysis with Self-Organizing Maps and Analytic

Provenance — D. Sacha, M. Kraus, J. Bernard, M. Behrisch, T. Schreck, Y. Asano, and D. A. Keim. IEEE Transactions on Visualization and Computer Graphics, 24(1), 120–130, 2018. doi: 10.1109/TVCG.2017.2744805

SOMFlow presents an analytical workflow for exploratory cluster analysis in which clustering is not treated as a single final result, but as a sequence of interpretable refinements. The research question is how analysts can iteratively discover meaningful structures in complex time series data while keeping track of the decisions that produced each clustering. The workflow begins with a dataset, such as pitch contours from speech recordings, and creates an initial self-organizing map. This first map gives an overview: similar time series are placed in nearby cells, each cell summarizes a subset of data, and visual encodings show representative contours, variation, density, quantization error, or metadata distributions.

The analyst first examines this initial map to understand whether it reveals meaningful clusters, artifacts, or relationships to metadata. For example, in the speech intonation studies, the analyst inspected whether pitch contour shapes were associated with speaker group, gender, utterance type, or repetition. The system supports this stage through multiple visual overlays: component planes show which time intervals drive differences, metadata coloring reveals whether certain cells are dominated by particular categories, and quality measures highlight cells that are internally heterogeneous or imprecise.

The second step is adaptation. If the map reflects unwanted structure, such as a gender-based pitch-height effect, the analyst modifies preprocessing or computation. In the study, the analyst applied domain-specific semitone normalization to remove the gender effect,

then generated a new SOM. Other adaptations include changing distance metrics, weighting parts of the time series, modifying SOM parameters, fixing cells, or changing normalization. The purpose is to steer the clustering toward structures relevant to the research question rather than artifacts of measurement or preprocessing.

The third step is partitioning. Once the analyst identifies interesting or problematic regions, SOMFlow supports creating new child maps from selected data subsets. The analyst can select individual cells, groups of cells, automatically detected clusters, or subsets defined by metadata values. For example, after identifying noisy cells caused by measurement errors, the analyst annotates and filters them out; after finding a relevant pitch-related pattern, the analyst splits the data by pitch condition or speaker group to compare the resulting subsets. Each new SOM provides a finer-grained view of a subset, allowing a divide-and-conquer analysis.

A defining feature of the workflow is that each partitioning or adaptation step becomes part of an explicit flow graph. The graph shows the provenance of the analysis: which SOMs came from which selections, filters, splits, or parameter changes. This enables reflection, comparison, and backtracking. The analyst can see the overall path from the original dataset through normalization, noise removal, metadata splits, and detailed subgroup comparisons. Linking and brushing across the graph shows where data items from one SOM appear in others, and color maps can be propagated across maps to compare structures.

Guidance is woven into the workflow. Quality and interestingness measures recommend imprecise cells to refine, metadata attributes likely to explain cluster structure, or similar neighboring cells that should be included in a selection. These recommendations do not replace the analyst's judgment but reduce the search space and help reveal promising next steps.

The analytical logic of SOMFlow is therefore cyclical: generate an overview clustering, inspect its quality and meaning, adapt computations if necessary, partition promising subsets, and use provenance to compare and reflect on the evolving analysis. This directly supports exploratory cluster analysis where the goal is not simply to produce clusters, but to understand how meaningful patterns emerge through iterative human-guided refinement.

45. SensePath: Understanding the Sensemaking Process

SensePath: Understanding the Sensemaking Process Through Analytic Provenance — P. H. Nguyen, K. Xu, A. Wheat, B. L. W. Wong, S. Attfield, and B. Fields. *IEEE Transactions on Visualization and Computer Graphics*, 22(1), 41–50, 2016. doi: 10.1109/TVCG.2015.2467611

SensePath describes a workflow for analyzing how people make sense of information during browser-based tasks. The research question is how analytic provenance can help qualitative researchers understand users' sensemaking processes more efficiently than by manually reviewing screen recordings and transcripts. The workflow begins with a participant performing an online sensemaking task, such as selecting accommodation for

a conference or choosing a product. While the participant uses a web browser, SensePath runs in the background and automatically captures low-level and mid-level provenance about the activity.

The first stage is data capture. SensePath records actions such as keyword searches, map searches, route searches, filtering operations, page revisits, link following, highlighting, annotation, and browsing. It also records timing, URLs, page titles, screenshots, search terms, filter parameters, and relationships between pages. The goal is not to fully infer the participant's thoughts automatically, but to capture enough structured evidence to support later qualitative interpretation. This capture is designed to be non-intrusive so that participants can perform the sensemaking task naturally.

The second stage is post hoc visual analysis by the researcher. SensePath presents the captured provenance in a timeline view, where actions appear in temporal order. The researcher begins by scanning this overview to identify broad phases of activity, recurring patterns, or moments that appear analytically interesting. For instance, a series of hotel pages opened from the same search site may suggest comparison, while repeated switches between a hotel page and a map may suggest location checking. The timeline can be zoomed, filtered, and aggregated, so the researcher can move between a full-session overview and detailed inspection of particular actions.

The next step is contextual interpretation. Selecting an action in the timeline opens the corresponding web page in a browser view, allowing the researcher to see what the participant saw at that moment. If the participant highlighted text or added annotations, the browser view can navigate to that material, giving additional evidence of what information mattered. A replay view is linked to the screen capture video and jumps automatically to the selected action's time. This lets the researcher inspect interactions not fully captured in provenance, such as scrolling, cursor movement, hesitation, or think-aloud comments. The workflow therefore combines structured action data with rich qualitative context.

The transcription view supports coding. It lists selected actions with start and end times, action types, automatically generated descriptions, and assigned themes. Researchers can use this material as a starting transcript and export it to qualitative analysis tools. The coding process involves assigning interpretive labels to action sequences, such as "finding conference location," "comparing hotel distances," "checking reviews," or "validating price." These codes can then be grouped into higher-level sensemaking steps or strategies.

The workflow proceeds from automatically captured provenance, to timeline-based overview, to contextual page and video inspection, to transcription and coding, and finally to reconstruction of a user's sensemaking process. In the evaluation, a researcher used SensePath to identify the sequence of steps and strategy followed by participants in an accommodation-selection task. The analytical logic is that low-level interaction traces do not directly reveal cognition, but they greatly reduce the burden of finding and interpreting the evidence from which sensemaking processes can be reconstructed.

46. Space-Time Visual Analytics of Eye-Tracking Data

Space-Time Visual Analytics of Eye-Tracking Data for Dynamic Stimuli — Kuno Kurzhals and Daniel Weiskopf. *IEEE Transactions on Visualization and Computer Graphics*, 19(12), 2129–2138, 2013. doi: 10.1109/TVCG.2013.194

This paper describes an analytical workflow for understanding how multiple viewers look at dynamic stimuli such as videos. The research question is how analysts can identify temporal patterns of attention, attentional synchrony, and dynamic areas of interest without watching the entire video repeatedly or manually defining areas of interest frame by frame. The workflow begins with eye-tracking recordings from multiple viewers watching the same video. These gaze data are synchronized with the video and combined with computer vision preprocessing, including optical flow estimation and shot boundary detection.

The first analytical step is overview. Instead of replaying gaze points over the video in real time, the system represents the video and gaze data in a space-time cube, where two dimensions correspond to screen space and the third corresponds to time. This static 3D representation lets the analyst see the whole temporal structure of attention at once. Dense bundles of gaze points indicate moments when viewers looked at the same region, while dispersed points indicate divided attention. Projection walls provide 2D summaries of the cube to reduce occlusion and support interpretation.

The analyst then interactively filters and inspects gaze distributions. In the data-point representation, gaze points are colored and made more or less visible according to their distance from the frame-wise center of mass. By adjusting a kernel-size parameter, sparse points fade out and dense regions remain. This allows the analyst to find episodes of attentional synchrony, such as moments when all viewers focus on a product label, a face, or a newly appearing object. These moments can then be interpreted in the context of the video frame.

The next step is spatiotemporal clustering. The system applies mean-shift clustering to gaze data, separated by detected shots so that cuts do not incorrectly merge unrelated gaze patterns. The resulting clusters are visualized as dynamic areas of interest, represented as axis-aligned boxes over the video and as hulls in the space-time cube. The analyst filters clusters by size to focus on regions that attracted many gaze points. This supports detection of multiple simultaneous areas of interest, such as viewers dividing attention among several objects in a scene. The analyst can compare cluster size, position, and duration to infer which objects drew more attention and how attention shifted over time.

Shot boundary analysis forms another part of the workflow. Automatically detected cuts are marked in the timeline and can be jumped to directly. Analysts inspect whether gaze remains at the previous location, recenters, or quickly shifts to a new salient object after the cut. This supports research questions about editing, cinematic guidance, and attentional reorientation.

The workflow also extends standard eye-tracking tools. Heat maps can be generated for selected time spans, and the paper introduces motion-compensated heat maps. These use optical flow to move gaze samples along with objects, so attention to a moving car or object accumulates on the object itself rather than being smeared across its path. Individual scan paths can also be shown for drill-down analysis.

Overall, the workflow moves from global spatiotemporal overview, to density filtering, to cluster-based area-of-interest discovery, to shot and object-level interpretation. It supports both exploratory discovery and detailed validation of attention patterns in dynamic stimuli, directly addressing the difficulty of analyzing group eye-tracking data for videos.

47. Guided Visual Exploration of Genomic Stratifications in Cancer (StratomeX)

Guided Visual Exploration of Genomic Stratifications in Cancer — M. Streit, A. Lex, S. Gratzl, C. Partl, D. Schmalstieg, H. Pfister, P. J. Park, and N. Gehlenborg. *Nature Methods*, 11(9), 884–885, 2014. doi: 10.1038/nmeth.3088

StratomeX describes an analytical workflow for exploring patient stratifications in cancer genomics. The research question is how analysts can efficiently compare many possible ways of grouping patients, relate those groupings to molecular and clinical data, and generate hypotheses about tumor subtypes. The workflow begins with a large cancer cohort containing heterogeneous data, such as mRNA expression, microRNA expression, protein expression, copy-number alterations, mutations, pathways, and clinical variables. From these data, many patient stratifications can be derived: clusters from molecular profiles, groups defined by gene alterations, sets based on clinical parameters, or subtypes produced by computational algorithms.

The first stage is visual comparison of selected stratifications. In the StratomeX view, each stratification is shown as a vertical column divided into stacked blocks, where each block represents a patient set. Bands connect patient sets across adjacent stratifications, and the width of each band shows how many patients overlap. The analyst uses this view to see how one grouping relates to another. For example, a cluster from expression data may overlap strongly with a mutation-defined group or a survival-defined clinical category. This supports the basic question of whether different data types tell consistent or complementary stories about tumor subtypes.

The next step is inspecting patient sets in context. Blocks can contain visual summaries such as heat maps, pathway maps with overlaid expression, survival plots, or other molecular and clinical views. The analyst compares patient sets not only by overlap but also by their associated biological and clinical characteristics. A promising subtype should be interpretable: it may have distinct survival, differential pathway regulation, characteristic mutations, or recognizable expression patterns. Thus the workflow connects set-level structure to molecular explanation.

Because the space of possible stratifications is too large to inspect manually, the workflow introduces query-based guided exploration. The analyst uses a query wizard to define what kind of relationship or property is being sought. A query may ask for stratifications similar to a selected stratification, stratifications that overlap with a particular patient set, patient groups that differ in survival, or pathways that are differentially regulated across sets. The system computes scores for relevant elements, such as stratifications, patient sets, clinical parameters, or pathways.

The results of these computations are shown in the LineUp view, a multi-attribute ranking visualization. The analyst examines ranked elements, adjusts filters and weights, and selects high-scoring candidates. Selecting an element in LineUp immediately visualizes it in StratomeX, allowing the analyst to interpret the computational result in the context of patient-set overlaps and molecular data. This tight loop between scoring, ranking, selection, and visual comparison is central to the workflow.

The process is iterative. The analyst begins with an initial stratification or patient set, queries the stratome for related or clinically meaningful structures, inspects ranked results, brings promising candidates into the StratomeX view, compares overlaps and data patterns, and then refines the query based on emerging hypotheses. In the renal cell carcinoma case study, this workflow supports characterization of tumor subtypes by linking molecular stratifications, clinical outcomes, and pathway-level differences.

The analytical logic is therefore a combination of broad computational search and detailed visual interpretation. StratomeX helps analysts move from thousands of possible patient groupings to a small number of biologically or clinically meaningful hypotheses, while preserving the ability to inspect how patient sets overlap and why a stratification may matter.

48. Tuner: Principled Parameter Finding for Image Segmentation Algorithms

Tuner: Principled Parameter Finding for Image Segmentation Algorithms Using Visual Response Surface Exploration — T. Torsney-Weir, A. Saad, T. Möller, B. Weber, H.-C. Hege, J.-M. Verbavatz, and S. Bergner. IEEE Transactions on Visualization and Computer Graphics, 17(12), 1892–1901, 2011. doi: 10.1109/TVCG.2011.248

Tuner describes an analytical workflow for finding good parameter settings for image segmentation algorithms. The research question is how to replace manual trial-and-error parameter tuning with a principled exploration of the parameter space. The workflow begins with a segmentation algorithm that has tunable parameters, a set of images with ground-truth segmentations, and one or more quality measures, such as Dice score, precision, recall, or physiological recovery error. Rather than manually selecting a parameter configuration, running the segmentation, checking the result, and guessing another configuration, Tuner treats the algorithm as an expensive black-box function whose response surface must be explored.

The first step is initial sampling of the parameter space. The user defines parameter ranges and chooses a sampling strategy, such as Latin hypercube sampling or random sampling.

Tuner generates a sparse but systematic set of parameter combinations. Each combination is passed to the segmentation algorithm, which runs independently and produces a segmentation. The segmentation is compared with the ground truth to compute one or more response values, representing segmentation quality. This stage can be performed as a batch process, since individual segmentations may be slow.

The second step is statistical modeling. Tuner fits a Gaussian process model to the sampled parameter-response pairs. This model estimates the response value at unsampled parameter configurations and also estimates uncertainty. The response surface is therefore not merely interpolated; the system can show where it is confident and where additional samples may be valuable. It also computes expected improvement, indicating regions where further sampling is likely to improve the best known result.

The analyst then enters the visual exploration stage. Tuner presents linked views for navigating the high-dimensional parameter space. A slice-based response view, inspired by HyperSlice, shows two-dimensional slices through the multidimensional response surface. The user can move through parameter values using sliders or direct interaction and can inspect how response values change locally. This supports understanding both optimality and sensitivity: broad stable regions are preferable because they suggest robust parameter settings, whereas narrow peaks indicate fragile configurations.

When multiple quality measures are involved, the workflow uses a Pareto-style view. The analyst examines trade-offs between response values, such as high Dice score versus low metabolic recovery error, or precision versus recall. Selecting a point in the trade-off plot moves the slice views to the corresponding parameter configuration. The analyst can then inspect nearby regions to see whether better or more stable trade-offs exist. This is important because the “best” segmentation may depend on several competing criteria, not a single scalar objective.

The next step is refinement. When the analyst identifies a promising or uncertain region, Tuner lets them mark a local hyperbox and place additional samples inside it. These new samples are run through the segmentation algorithm, response values are computed, and the Gaussian process model is updated. The workflow cycles between model inspection, region selection, additional sampling, and reassessment until the analyst is confident in a parameter range.

The paper demonstrates this workflow on two segmentation problems: dynamic PET brain imaging and microtubule extraction from electron tomograms. In both cases, the goal is not only to find a high-scoring parameter point, but to understand stable regions that generalize across images. The analytical logic is thus systematic response-surface exploration: sample broadly, model uncertainty, inspect trade-offs, refine promising regions, and choose robust parameter settings grounded in evidence rather than guesswork.

49. VBridge: Connecting the Dots Between Features and Data to Explain Healthcare Models

F. Cheng, D. Liu, F. Du, Y. Lin, A. Zyttek, H. Li, H. Qu and K. Veeramachaneni. "VBridge: Connecting the Dots Between Features and Data to Explain Healthcare Models." IEEE Transactions on Visualization and Computer Graphics, vol. 28, no. 1, pp. 378–388, Jan. 2022. doi: 10.1109/TVCG.2021.3114836

The analytical workflow in VBridge is designed around a central clinical question: when a machine-learning model predicts a health risk for a patient, how can clinicians understand whether that prediction is trustworthy, clinically meaningful, and supported by the patient's actual medical record? The workflow begins before visualization, with electronic health record data being transformed into machine-learning features. Because such engineered features often summarize raw clinical events—such as the mean, standard deviation, or trend of a vital sign over a time interval—the system preserves traceable links between each feature and the underlying records from which it was derived. The model then produces predictions, and SHAP values are computed to estimate how much each feature contributed to a particular patient-level prediction.

Once a clinician starts analysis, the first step is to select the patient, prediction target, and a relevant reference cohort. This cohort provides the clinical baseline against which the patient's feature values and raw observations can be interpreted. The clinician then reviews a high-level profile of the patient and a timeline overview of the medical record, gaining situational context before engaging deeply with model explanations. This step matters because the paper's research question is not simply how to display feature importance, but how to embed explanation into clinical reasoning.

The workflow can proceed in two directions. In a forward workflow, typically preferred by senior clinicians, the clinician first inspects the patient's raw records, forms an independent clinical hypothesis, and then compares this hypothesis with the model's prediction and feature-level explanation. If the model emphasizes the same risk factors that the clinician expected, trust in the prediction increases. If the explanation diverges from expectation, the clinician investigates the corresponding raw records to determine whether the model has identified a plausible but overlooked signal or whether the prediction should be rejected.

In a backward workflow, the clinician begins from the model output and its explanation. The feature view ranks and groups features hierarchically, allowing the clinician to move from broad clinical categories to specific engineered quantities. Each feature is shown together with its contribution to the prediction and its reference range within the selected cohort. The clinician can filter, sort, and expand feature groups to identify the factors most responsible for the prediction. If a highly contributing feature has an abnormal value, the clinician may perform a what-if analysis, replacing the abnormal value with a reference-range value to see whether the prediction or contribution would substantially change.

The key bridging step follows: the clinician moves from abstract feature explanations back to original patient records. Linked temporal views show the raw time series behind selected

features, highlight abnormal observations relative to cohort references, and mark periods that most influenced a feature. This allows the clinician to ask not only “which feature mattered?” but “what actually happened to the patient, when, and in what clinical context?” If necessary, the clinician returns to the broader timeline to inspect concurrent labs, vital signs, or events. The workflow therefore cycles between prediction, feature explanation, cohort evidence, and raw records until the clinician can decide whether the model’s reasoning supports clinical action.

50. Visual Movie Analytics workflow

Kuno Kurzhals, Markus John, Florian Heimerl, Paul Kuznecov, and Daniel Weiskopf. “Visual Movie Analytics.” *IEEE Transactions on Multimedia*, vol. 18, no. 11, pp. 2149–2160, 2016. doi: 10.1109/TMM.2016.2614184

The analytical workflow in “Visual Movie Analytics” addresses the question of how researchers can analyze the semantic and structural content of a full-length film without relying only on sequential viewing and manual note-taking. The workflow begins by treating a movie as a combination of video content, subtitles, and script text. The video stream provides visual structure and motion information, while the script supplies semantic information that is usually absent from the image data alone, such as scene descriptions, character names, locations, and narrative actions. The purpose of the workflow is to bridge these levels so that analysts can answer questions about who appears, what happens, where events occur, and when they occur in the film.

The first phase is automatic preprocessing. On the image side, the system analyzes optical flow to detect shot boundaries and to derive motion signatures that can later be used for visual similarity search. On the textual side, subtitles are parsed as time-stamped dialogue, while the movie script is parsed into scene headings, action descriptions, character names, and dialogue. Because scripts are not inherently synchronized with the final movie, the workflow aligns script dialogue with subtitle text. Once script passages are matched to subtitles, their timestamps are used to map semantic scene information onto the actual video. This alignment turns the movie into a temporally organized structure of scenes, shots, characters, actions, and semantic descriptions.

After preprocessing, the analyst enters an interactive visual analytics environment built around multilayer timelines. The first layer provides an overview of the complete movie, with separate timelines for extracted elements such as characters, groups, keywords, motion patterns, or semantic-frame similarities. This overview allows the analyst to see, for example, where a character appears or where a theme recurs across the film. When the analyst selects a scene, a second layer expands that scene into its constituent shots, making local structure visible. A third layer can display sampled video frames as a storyboard, linking abstract patterns back to concrete audiovisual evidence.

The analytical reasoning then proceeds iteratively. The analyst may begin with an exploratory question, such as which characters are central to the plot, and inspect occurrence patterns across the overview. Alternatively, the analyst may formulate a query, such as finding all scenes involving a specific character pair, a keyword, a semantic frame,

or motion pattern. Query results become new visual elements in the timeline, so they can be compared with existing character, scene, and topic information. The analyst can also search for scenes similar to a reference scene using motion, text similarity, or semantic-frame similarity. This supports reasoning from known examples to structurally or semantically related moments elsewhere in the film.

A key part of the workflow is annotation. When the analyst identifies meaningful events, such as battles, imprisonments, or scenes involving an important object, these can be marked as new elements and reused in later queries. Thus, the workflow moves from automatic extraction to human interpretation, then back into computational search. In the Lord of the Rings case study, this process supports analysis of character constellations, major plot events, battle scenes, and the occurrence of the ring. The workflow's analytical logic is therefore cumulative: low-level video and text features provide entry points, interactive visual comparison helps identify meaningful patterns, and analyst annotations gradually build higher-level interpretations of the movie's narrative and stylistic structure.

51. Detecting Stable Cross-Impact Patterns in Bivariate Time Series

G. Andrienko, N. Andrienko, M. Akila, B. Kathirgamanathan and M. Ponce-de-Leon.
"Detecting Stable Cross-Impact Patterns in Bivariate Time Series." IEEE Transactions on Visualization and Computer Graphics, vol. 32, no. 7, pp. 5996–6013, 2026. doi: 10.1109/TVCG.2026.3676810

The workflow in this paper addresses the problem of finding stable lagged relationships between pairs of time series. The research question is not merely whether two time series are correlated, but whether changes in one series are followed, after varying delays, by consistent changes in another series, and whether such patterns recur across many time series pairs. The workflow begins with multiple bivariate time series, such as population mobility and COVID-19 incidence for different provinces, or financial indicators such as gold and oil prices. One series is treated as the base series and the other as the response series, reflecting the assumption that the response may react after a delay.

The first analytical step is to compute cross-impact values. For each pair of time series, the method slides a time window along the base series and compares it with time-shifted windows in the response series over a range of possible lags. Several association measures can be used, but the paper introduces a tolerance-based variant of Kendall's tau. This variant treats small fluctuations as ties rather than meaningful increases or decreases, making the measure less sensitive to noise. The result is a two-dimensional matrix of impact values indexed by absolute time and lag. Each cell indicates whether a segment of the base series is followed by a similar or opposite trend in the response series after a specific delay.

The second step is visual calibration using the Ikat plot. This plot displays the impact matrix as a colored time-lag field: red for positive impacts, blue for negative impacts, and pale colors for weak or absent impacts. The analyst examines how patterns change with parameter choices such as window length, lag range, tolerance threshold, trend threshold,

and impact-strength threshold. Vertical stripes, diagonal stripes, isolated hotspots, and larger coherent regions correspond to different types of lagged relationships, such as sustained impacts, transient impacts, or stable trend-driven impacts. The Ikat plot is linked to the original time series, so the analyst can inspect the actual base and response fragments behind any matrix cell. Dynamic queries further restrict the view to periods with specified value levels, trends, lags, or impact strengths. This step is crucial because the workflow depends on selecting parameters that reveal meaningful patterns rather than noise.

Once parameters are selected from representative examples, the third step is event extraction. The system applies the chosen criteria to all time series pairs and extracts contiguous regions in the time-lag matrix as impact events. Each event has a time interval in the base series, a lag interval in the response series, and attributes describing its duration, depth, trend direction, and impact strength. This transforms abstract statistical relationships into discrete temporal objects that can be analyzed like events.

The fourth step analyzes the distribution and relationships of these events. In the COVID-19 case study, the authors also extract trend events, such as increasing cases, decreasing cases, increasing trips, and decreasing trips. Timeline visualizations show how trend and impact events are distributed across provinces and time, revealing waves, regional differences, and periods where expected impacts do or do not occur. The workflow then quantifies whether trends were impacted by previous events or had impacts on later events, and logistic regression models are used to examine which event features predict subsequent influence.

The analytical logic therefore moves from raw time series, to robust lagged impact measurement, to visual parameter discovery, to event extraction, and finally to temporal, spatial, and statistical analysis of event patterns. This enables experts to study complex delayed relationships without requiring strong causal assumptions.

52. RuleSense: Interactive Visual Exploration of Rule-Based Model Logic

N. Andrienko, G. Andrienko, and B. Kathirgamanathan. "Interactive Visual Exploration of Rule-Based Model Logic." *IEEE Transactions on Visualization and Computer Graphics*, 2026, early access. doi: 10.1109/TVCG.2026.3689736

The workflow in RuleSense is designed to answer a model-auditing question: does a rule-based machine-learning model reason in ways that are logically consistent, domain-plausible, and trustworthy, beyond simply achieving good predictive accuracy? The workflow begins with a trained model that can be represented as decision rules. In the paper's main examples, random forests are converted into rule sets by extracting root-to-leaf paths from all trees. This produces thousands of rules, each combining feature conditions with a predicted class or value. Because such rule sets can contain redundancy, contradictions, and overly specific conditions, the workflow treats the model itself as the object of analysis.

The first step is to create a global overview of the rule set. Features are divided into value intervals, and heatmap-based displays show how often feature intervals occur in rules for each predicted class. This overview helps analysts see which features the model uses to recognize each outcome and whether the value ranges make sense. For example, in the vessel movement case study, anchoring rules are expected to involve low speeds and proximity to ports, while forward movement should involve higher speeds and strong movement away from the starting point. The overview allows domain experts to detect whether such expectations are broadly reflected in the model logic.

The second step is interactive filtering and drill-down. Analysts restrict the rule set by predicted class, feature presence or absence, or feature-value intervals. This supports hypothesis-driven inspection of the model's logic. For instance, an analyst can ask whether rules predicting port entry or exit include distance-to-port features, or whether rules predicting anchoring impose an upper bound on speed. Rules that omit essential features or use implausible value ranges become candidates for closer inspection. Importantly, this part of the workflow can operate even without labeled data, because the analyst is evaluating the semantics of the rules themselves rather than their performance on cases.

The third step applies computational rule analysis. Contradiction detection identifies cases where a more general and a more specific rule predict different outcomes, suggesting unstable or ambiguous model logic. Subsumption detection identifies rules that are more specific than other rules with the same outcome and may therefore be redundant. Similarity-based grouping and generalization help analysts reason about clusters of related rules rather than individual rules one by one. These computations do not automatically rewrite the model; they guide expert attention to parts of the logic that may need review.

If labeled data are available, the workflow can continue to rule testing and refinement. Analysts can remove or modify suspicious rule subsets and evaluate whether predictive accuracy changes. In the vessel movement case, logically questionable rules were removed with little or no accuracy loss, showing that the model could be simplified and made more domain-consistent without degrading performance.

A further step explores higher-order feature relationships using topic modeling. Rules are treated as documents and encoded feature conditions as terms. Nonnegative matrix factorization then reveals recurring groups of conditions, or "topics," that represent latent decision patterns. Analysts inspect topic-feature matrices, topic weights, and class associations to understand which condition combinations recur in the model and how they relate to outcomes. In the COVID-19 case study, this revealed that the model relied primarily on recent disease levels, while mobility features played a more contextual and time-dependent role.

Overall, the RuleSense workflow moves from global rule-set structure, to focused filtering, to contradiction and redundancy analysis, to topic-based discovery of feature interactions, and optionally to performance-aware refinement. Its analytical logic is centered on model reasoning rather than prediction output: it helps experts ask whether the model's internal rules constitute a plausible representation of the domain.

53. Understanding Compound Potency Models

B. Kathirgamanathan, T. Janela, E. Xerxa, G. Andrienko, J. Bajorath and N. Andrienko.
"From Prediction to Insight: Visual Analytics for Understanding Compound Potency Models." *IEEE Computer Graphics and Applications*, vol. 46, no. 3, pp. 133–140, 2026.
doi: 10.1109/MCG.2026.3675766

The workflow in this paper addresses a medicinal-chemistry question: how can a machine-learning model for compound potency be used not only to predict activity, but to reveal chemically meaningful combinations of molecular features associated with high potency? The workflow begins with a dataset of compounds acting on MAPK p38 alpha kinase, each represented by extended connectivity fingerprints. These fingerprints encode the presence or absence of local molecular fragments as binary features. Rather than building a maximally complex predictor, the authors select a limited set of relevant fingerprint features and train a random-forest regression model whose internal logic can later be converted into decision rules.

The first analytical step after model training is rule extraction. Each path from the root to a leaf in each decision tree becomes a rule, specifying a combination of fragment-presence or fragment-absence conditions and a predicted potency value. This transformation is central to the workflow because it turns the model from an opaque ensemble into a large collection of explicit structural hypotheses: if certain fragments are present or absent together, the model predicts a certain potency. The resulting rule set is too large for direct inspection, so visual analytics is used to move from the full rule collection toward interpretable subsets.

The second step provides an overview of feature–prediction relationships. Predicted potency values are divided into low, medium, and high intervals, and heatmap summaries show how frequently each feature-value condition appears in rules belonging to each interval. Analysts use this overview to identify fragments that occur disproportionately in high-potency rules or whose absence appears important. The goal is not to claim that any single fragment determines potency, but to identify promising starting points for exploring interactions. Interactive filtering then lets analysts select rules containing specific conditions, such as feature 266 being present, and observe how the distribution of predicted potency changes. When adding another condition, such as feature 145 being present, further enriches the rule subset for medium and high potency, this suggests a possible synergistic fragment combination.

Because exhaustive manual exploration of all feature combinations is infeasible, the third step applies topic modeling to the rule set. Each rule is treated as a document and each feature-value condition as a term. Nonnegative matrix factorization identifies topics, each representing a group of conditions that often co-occur across rules. The authors compare different numbers of topics using embedding coverage, dominant topic strength, and the ability to separate high-potency rules. A topic–condition matrix shows which fragment conditions define each topic, while a parallel-coordinates view shows how strongly individual rules are associated with topics and how this relates to predicted potency. A

two-dimensional projection of topic-weight vectors then helps isolate clusters enriched in high-potency rules.

The fourth step zooms into the high-potency cluster. After selecting this region, redundant rules are pruned so that analysis focuses on distinct logical patterns. Topic modeling is reapplied within the selected subset to discover more specific combinations that distinguish high-potency rules from low and medium ones. Analysts then use topic-guided filtering to refine candidate feature combinations. For example, combinations involving the presence of features 266 and 164 and the absence of features 34 and 27 consistently lead to high predicted potency. This stage converts broad co-occurrence patterns into concrete candidate rules about structural fragments.

The final step maps these fingerprint conditions back to molecular substructures. Chemists inspect the corresponding fragments, such as chlorobenzene, piperidine, quinolone-related structures, and other ring systems, and compare them against highly potent training and test compounds. The identified combinations are validated by screening test compounds and checking whether compounds matching the patterns receive accurate high-potency predictions. Thus, the workflow progresses from trained model, to decision rules, to feature-frequency overview, to topic-guided rule clustering, to candidate fragment combinations, and finally to chemically interpretable molecular motifs. Its analytical logic is to transform predictive regularities into testable structure–activity hypotheses.