

AI-Assisted Design of Visual Analytics Workflows for Bike-Sharing Data Analysis

Session Report

1 Introduction

This document records an interactive session in which a large-language-model assistant was used to design a visual analytics (VA) workflow for analysing spatio-temporal patterns in public bike-sharing data. The session proceeded in several steps: the assistant was first provided with a workflow description language and a library of published VA workflows; the user then described the analytical goals; and the assistant proposed, refined, and documented a complete workflow, drawing on relevant examples from the library.

Each subsection below corresponds to one conversational turn. The user’s prompts are quoted verbatim in shaded boxes; the assistant’s responses are summarised in the text that follows.

2 Step 1: Providing the Workflow Language Definition

User prompt:

The attached document contains a definition of a language called ATWL designed for representation of visual analytics workflows. You need to understand the language concepts and semantics it is meant to express, and also the syntax used for expressing the semantics. In the next step, you will be given a “library” of VA workflows extracted from published papers. Your task will be to help a user to design a workflow for achieving user’s goals by finding relevant examples in the “library” and suggesting how they can be adapted to the user’s case.

The assistant confirmed its understanding of the ATWL language by summarising its main concepts:

- **Eight artifact types:** entities, feature, arrangement, visualisation, pattern, model, knowledge, and specification.
- **Eight transform intents:** define-unit, characterise, contextualise, visualise, abstract, build-model, generate-knowledge, and assess.
- **Control structures:** loops with implicit or explicit termination conditions, conditionals, and assignment for iteration.
- **Workflow structure:** a header with an optional template summarising analytic stages, forming a directed acyclic dependency graph (with explicit loop assignments as the only exception).

3 Step 2: Providing the Workflow Library

User prompt:

I have attached an additional file with the workflow library. You can perform initial analysis for revealing reusable and adaptable parts.

The assistant performed a systematic analysis of all 17 workflows in the library. The results were organised along several dimensions.

Workflow index. A table cataloguing each workflow by data type and primary analytical goal was produced, covering domains from regular time series and movement trajectories to text documents and trained machine-learning models.

Reusable building blocks. Eight recurring structural blocks were identified across the library:

- A. **Temporal partitioning into episodes** — segmenting time series into fixed or domain-driven windows (found in Workflows 1.1, 1.2, 1.9, 1.12).
- B. **Iterative clustering with visual assessment** — the single most common block, appearing in eight workflows. A loop of clustering, summarisation, visualisation, and parameter adjustment.
- C. **Dimensionality-reduction projection** — characterising entities as vectors, projecting into 2D, and visualising as a scatterplot (Workflows 1.2, 1.8, 1.12, 1.15).
- D. **Iterative model building and refinement** — fitting a model, evaluating through residual or prediction analysis, and refining (Workflows 1.10, 1.11, 1.12, 1.13).
- E. **Multi-level interactive exploration** — navigating, filtering, and drilling down through levels of abstraction (Workflows 1.14, 1.15, 1.16, 1.17).
- F. **Feature engineering loop** — iteratively refining feature computation within the analytical loop (Workflow 1.12).
- G. **Spatio-temporal aggregation** — aggregating by place and time interval (Workflows 1.3, 1.6, 1.11).
- H. **Terminal knowledge synthesis** — the final abstraction and knowledge-generation step present in all 17 workflows.

Structural archetypes. Five workflow archetypes were identified: (I) Cluster–Contextualise–Explore–Interpret; (II) Progressive Refinement through Successive Questions; (III) Iterative Model Building and Validation; (IV) Model Understanding and Diagnostics; (V) Simplify–Explore–Recommend.

Cross-cutting patterns. Several patterns spanning multiple workflows were noted, including dual-loop refinement, specification-driven iteration, coordinated multi-view assessment, progressive abstraction, hypothesis-driven verification, residual-based refinement, and two-stage clustering.

Quick-reference table. A mapping from common user goals (e.g., “find temporal patterns,” “build a predictive model,” “diagnose an existing ML model”) to relevant library workflows and building blocks was provided.

4 Step 3: Describing the Analytical Goals

User prompt:

I have data about use of public bikes in a city. The records represent trips: bike ID, origin place and time, destination place and time. The origin and destination places are docking stations distributed over the city. The coordinates and capacities of the docking stations are given. In many popular places, there are multiple docking stations clustered close to each other. I need to analyse the spatio-temporal patterns of taking and returning bikes. I am especially interested in revealing patterns of stations running out of available bikes and stations overcrowded with returned bikes. The patterns are expected to be related to the diurnal and weekly cycle as well as seasons and holidays. I think, the analysis should be done not at the level of individual stations but at the level of “places” containing groups of spatially close stations. Based on my analysis, I want to develop a model that will tell me for any given date and time of the day how bikes should be allocated/relocated in order to satisfy the demands and avoid overfilling of stations.

The assistant identified the most relevant library workflows for each aspect of the user’s goals (Table 1) and produced a complete workflow specification in the ATWL language. The workflow, fully shown in Appendix A, was organised into six phases:

Phase 0: Define places by spatially clustering docking stations.

Phase 1: Aggregate trip data into place-level time series and compute availability.

Phase 2: Discover temporal demand-supply patterns by clustering days and visualising results on a calendar.

Phase 3: Identify and analyse critical events (shortages and overcrowding).

Phase 4: Build predictive demand-supply models with residual-based refinement.

Phase 5: Develop and validate an allocation/relocation strategy through simulation.

A concluding Phase 6 synthesises all findings into actionable knowledge. Table 1 summarises which library workflows and building blocks informed each phase.

Table 1: Mapping from user goals to library workflows and reusable blocks.

Phase	User Goal	Sources	Adapted Blocks
0	Group stations into places	1.6, 1.3	Spatial clustering loop
1	Aggregate to place-level series	1.11, 1.6	Spatio-temporal aggregation
2	Discover diurnal/weekly/seasonal patterns	1.1, 1.3, 1.9	Day profiling, clustering, calendar
3	Reveal shortage & overcrowding	1.6	Event extraction, contextualisation
4	Build predictive model	1.11, 1.10	Residual-based model refinement
5	Develop relocation strategy	1.5	Prescriptive plan-tuning loop

5 Step 4: Natural-Language Workflow Description

User prompt:

This is great. Now assume that the user is not familiar with ATWL. Represent the recommended workflow in natural language. Include references to papers where the user can see relevant examples and specify which fragments or aspects of the referred workflows are relevant.

The assistant produced a detailed natural-language description of the six-phase workflow, written for a reader with no knowledge of ATWL. Each phase was presented as a subsection with two

parts: *What you do* (step-by-step instructions in plain English) and *Where to find relevant examples* (pointers to specific published papers with annotations of which fragments are most relevant).

The key elements of the description are summarised below.

Phase 0 — Defining places. Spatially cluster nearby docking stations into “places” using a distance-threshold algorithm; display results on a map; iteratively adjust parameters until places are spatially coherent and at an appropriate granularity. The iterative spatial-clustering loop with visual assessment on a map follows the approach of Workflow 1.6 [1], while the idea of using quality metrics to guide aggregation parameters is drawn from Workflow 1.3 [7].

Phase 1 — Aggregation. For each place and hourly interval, count bike takes and returns, compute net flow, directed inter-place flows, running availability, and occupancy rate. This spatio-temporal aggregation step producing one time series per spatial unit follows Workflow 1.11 [2]; the additional aggregation of inter-place flows follows Workflow 1.6.

Phase 2 — Temporal pattern discovery. Partition all place-level time series into daily episodes; represent each day by its hourly demand-supply profile across all places; hierarchically cluster days by profile similarity; display clusters on a calendar view coordinated with profile line graphs, place-level heatmaps, and flow-map thumbnails. Interpret and label clusters (e.g., “regular weekday,” “summer weekend,” “public holiday”). Iterate on the number of clusters and distance measure until the decomposition is clear and interpretable.

The core logic of day clustering with a calendar view and profile line graphs comes directly from Workflow 1.1 [6]. The extension to flow data with calendar-positioned thumbnails is adapted from Workflow 1.3. The progressive-abstraction strategy for multivariate temporal data draws on Workflow 1.9 [3].

Phase 3 — Critical events. Define shortages and overcrowding as episodes where occupancy crosses domain-defined thresholds; extract all such episodes; aggregate by place, day type, and hour; visualise on a map with glyphs and on a calendar heatmap; interpret spatial complementarity (e.g., morning shortage at residential places paired with overcrowding at business-area places). The event-extraction and spatio-temporal-aggregation approach follows Workflow 1.6.

Phase 4 — Predictive modelling. Group places by demand-similarity; identify temporal components (diurnal, weekly, seasonal, holiday); fit seasonal time-series models with multiple cycles; overlay model curves on data; iteratively refine via residual analysis. The full modelling loop—grouping, representative derivation, model fitting, visual comparison, and residual-based evaluation—follows Workflow 1.11. The structured residual-based feature-discovery loop is complemented by insights from Workflow 1.10 [5].

Phase 5 — Allocation strategy. Formulate an optimisation model for bike distribution given predicted demand; validate by simulating recommendations on historical dates sampled from each day-type cluster; visualise actual vs. recommended critical-event counts; iterate on allocation criteria until improvement is satisfactory. The prescriptive plan-specification, impact-estimation, and refinement cycle follows Workflow 1.5 [4].

Phase 6 — Knowledge synthesis. Integrate all findings: named demand-supply pattern types and their calendar distribution; critical-event patterns with causal flow explanations; predictive models with documented accuracy; and validated allocation recommendations with operational guidance.

6 Step 5: L^AT_EX Formatting

User prompt:

Transform this text to Latex format to be included as a section in an existing Latex document.

The assistant converted the natural-language workflow description into a L^AT_EX section with subsections for each phase, using `\paragraph{}` for sub-steps, `itemize` and `enumerate` environments for lists, a summary table (`tabularx` with `booktabs`), and `\cite{}` commands for all paper references. Suggested Bib_TE_X entries for all cited works were also provided.

7 Step 6: Workflow Diagram

User prompt:

It would be good to generate also a graphical representation of the workflow as a flow diagram, perhaps, at a higher abstraction level to facilitate readability and understanding. For example, it may be a figure generated using tikz package of Latex. From my prior work, I have the following Latex definitions, which you can use if suitable:

```
\tikzset{
  artifact/.style={rectangle, draw=black, thick,
    fill=blue!10, ...},
  transform/.style={ellipse, draw=black, thick,
    fill=green!10, ...},
  human/.style={artifact, fill=orange!10,
    line width=1.5pt},
  loop/.style={draw, dashed, inner sep=0.5cm, ...},
  arrow/.style={-Stealth, thick}
}
```

The assistant generated a full-page TikZ figure presenting the workflow at a high abstraction level. Green ellipses represent computational transforms, bold-bordered orange rectangles represent human assessment steps, and blue rounded rectangles represent data artifacts passed between phases. Dashed borders and feedback arrows indicate iterative phases. Phases 3 and 4 were placed side by side to convey their structural independence.

8 Step 7: Diagram Refinement

User prompt:

The screenshot shows how the figure looks like in a compiled PDF. Please improve the placements of labels and arrows to avoid overlapping and slanted arrows where they could be straight.

Based on a screenshot of the compiled figure, the assistant identified and corrected several layout issues:

- **Overlapping phase labels:** labels were moved from external anchors on the box border to dedicated nodes *inside* the fitted region, ensuring automatic enclosure with adequate padding.
- **Slanted branch arrows** (Phase 2 output to Phases 3 and 4): replaced with orthogonal routing using a vertical connector to a branch-point coordinate, followed by horizontal-then-vertical paths (-|).
- **Slanted merge arrows** (Phases 3/4 outputs to Phase 5): replaced with orthogonal routing using vertical-then-horizontal paths (-|) converging at a merge-point coordinate.
- **Junction indicators:** filled dots added at branch and merge points.
- **Feedback arrows clipped by box borders:** bottom phantom coordinates pushed further below the node row to ensure loop-back paths are fully enclosed.
- **Cramped vertical spacing:** inter-phase gaps increased throughout.
- **Parallel phases too close:** horizontal centres of Phases 3 and 4 widened from ± 3.15 to ± 4.8 units.

The corrected diagram is shown in Figure 1.

9 Observations

Several aspects of the session are worth noting.

1. **Reuse of library patterns.** The final workflow draws on six of the 17 library workflows. No single library workflow covers the user’s needs, but the combination of building blocks from multiple workflows—spatial clustering (Workflow 1.6), day-profile clustering with calendar views (Workflow 1.1), temporal clustering of flow data (Workflow 1.3), spatio-temporal modelling with residual evaluation (Workflow 1.11), residual-based feature discovery (Workflow 1.10), and prescriptive plan-tuning (Workflow 1.5)—yields a coherent end-to-end workflow.
2. **Adaptation, not copying.** In each case, borrowed fragments required non-trivial adaptation: the spatial clustering loop from Workflow 1.6 was simplified (station locations are given, not extracted); the day-profiling idea from Workflow 1.1 was extended to multi-place matrices; the modelling loop from Workflow 1.11 was augmented with holiday and multi-scale seasonal components; and the prescriptive loop from Workflow 1.5 was transposed from event-sequence recommendations to fleet-allocation optimisation.
3. **Iterative refinement of outputs.** The session itself was iterative: the formal ATWL specification was first produced, then translated into natural language with paper references, converted to \LaTeX , augmented with a diagram, and the diagram was corrected after visual inspection of compiled output. This mirrors the human-in-the-loop refinement loops present in the workflow itself.
4. **Diagram as communication aid.** The high-level flow diagram (Figure 1) abstracts away the detailed artifact dependencies of the ATWL specification and focuses on the phase structure, iteration loops, and data flow. It serves a complementary role to the textual description: the text explains *what* and *why*; the diagram shows *how the parts connect*.

References

- [1] Gennady Andrienko, Natalia Andrienko, Christophe Hurter, Salvatore Rinzivillo, and Stefan Wrobel. From movement tracks through events to places: Extracting and characterizing significant places from mobility data. In *2011 IEEE Conference on Visual Analytics Science and Technology (VAST)*, pages 161–170, 2011.

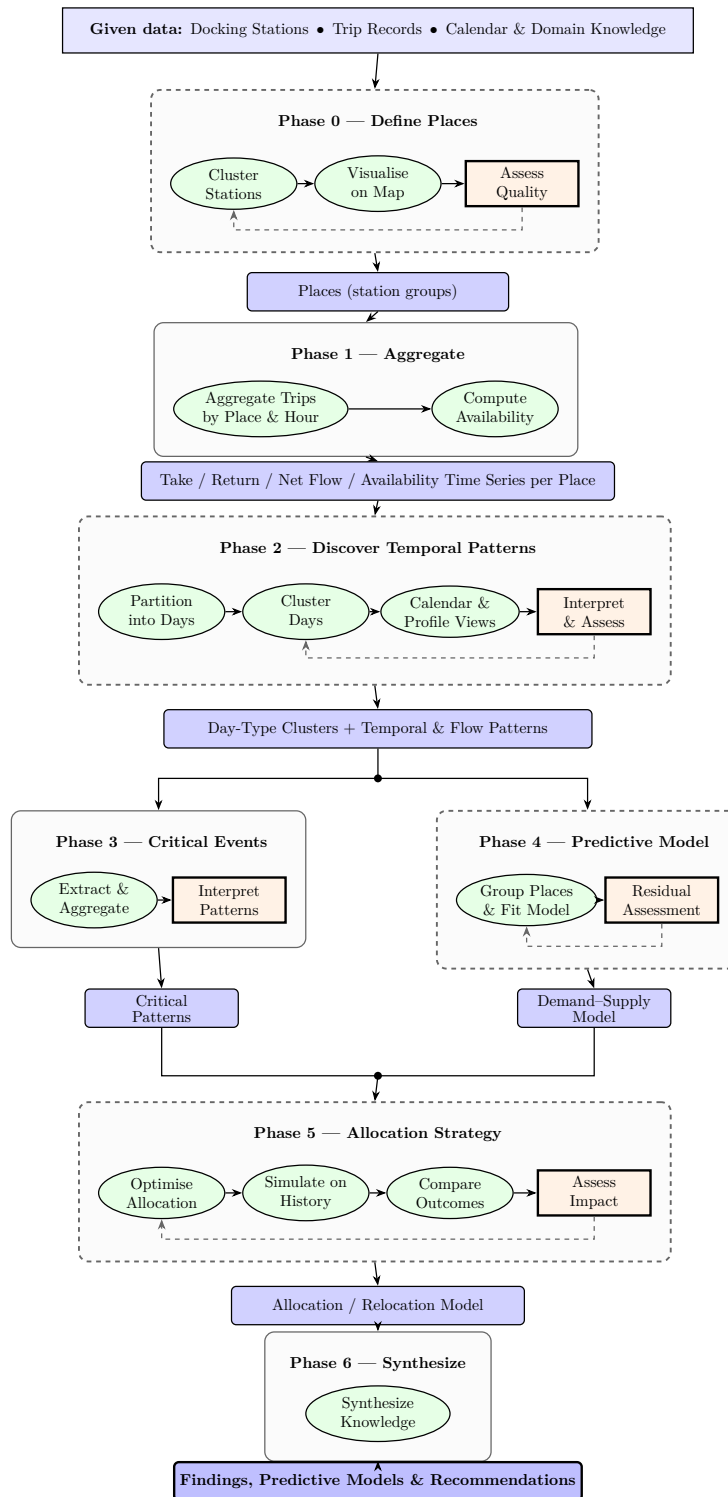


Figure 1: High-level workflow for analysing bike-sharing spatio-temporal patterns. Green ellipses represent computational transforms; bold-bordered orange rectangles represent human assessment and decision steps; blue rounded rectangles represent data artifacts passed between phases. Dashed phase borders and dashed feedback arrows indicate iterative loops with human-in-the-loop refinement. Phases 3 and 4 are independent and may proceed in parallel; their outputs converge into Phase 5. Black dots mark branch and merge junctions.

- [2] Natalia Andrienko and Gennady Andrienko. A visual analytics framework for spatio-temporal analysis and modelling. *Data Mining and Knowledge Discovery*, 27(1):55–83, Jul 2013.
- [3] Natalia Andrienko, Gennady Andrienko, and Gota Shirato. Episodes and topics in multivariate temporal data. *Computer Graphics Forum*, 42(6):e14926, 2023.
- [4] Fan Du, Catherine Plaisant, Neil Spring, and Ben Shneiderman. Eventaction: Visual analytics for temporal event sequence recommendation. In *2016 IEEE Conference on Visual Analytics Science and Technology (VAST)*, pages 61–70, 2016.
- [5] Thomas Mühlbacher and Harald Piringer. A partition-based framework for building and validating regression models. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):1962–1971, 2013.
- [6] Jarke J. van Wijk and Edward R. van Selow. Cluster and calendar based visualization of time series data. In *Proceedings of the IEEE Symposium on Information Visualization (InfoVis '99)*, pages 4–9, Los Alamitos, CA, USA, 1999. IEEE Computer Society.
- [7] Tatiana von Landesberger, Felix Brodkorb, Philipp Roskosch, Natalia Andrienko, Gennady Andrienko, and Andreas Kerren. Mobilitygraphs: Visual analysis of mass mobility dynamics via spatio-temporal graphs and clustering. *IEEE Transactions on Visualization and Computer Graphics*, 22(1):11–20, 2016.

A Recommended Workflow for Analysing Bike-Sharing Spatio-Temporal Patterns

A.1 Overview

The analysis proceeds through six phases, each building on results from the previous one. The overall logic is:

1. **Define “places”** by grouping nearby docking stations.
2. **Aggregate trip data** to the place level and compute availability time series.
3. **Discover temporal demand-supply patterns** and relate them to the weekly, seasonal, and holiday calendar.
4. **Identify and analyse critical events**—shortages and overcrowding.
5. **Build predictive models** of bike demand and supply for each place.
6. **Develop and validate an allocation/relocation strategy.**

Below, each phase is described in detail, along with pointers to published workflows that illustrate similar analytical steps.

A.2 Phase 0: Defining Places from Docking Stations

A.2.1 What You Do

You start with the known locations and capacities of all docking stations. Because many popular areas have multiple stations clustered close together, you group spatially nearby stations into *places*—your primary spatial units for the entire analysis.

Use a spatial clustering algorithm (for example, density-based clustering or distance-threshold grouping) to assign each station to a place. For each place, compute the **total docking capacity** (sum of member station capacities) and a **centroid location**.

Display the result on a **geographic map**: stations coloured by place membership, with place boundaries drawn around each group. Visually assess whether the grouping makes sense—are popular areas with multiple stations properly unified? Is the granularity appropriate (not too coarse, not too fine)? If not, adjust the distance threshold or minimum group size and repeat.

This is an **iterative loop**: cluster → visualise on map → assess → adjust parameters → re-cluster, until you are satisfied with the places.

A.2.2 Where to Find Relevant Examples

- **Workflow 1.6** [1] demonstrates how to delineate meaningful places from data through iterative spatial clustering with visual assessment. In that paper, places are derived from movement events using density-based clustering with a custom distance function, and the analyst iteratively adjusts clustering parameters while inspecting results on a map and in a space-time cube. *Relevant aspects*: the iterative loop of clustering → map visualisation → quality assessment → parameter adjustment. Your case is simpler because station locations are already given (you do not need to extract events first), but the iterative refinement logic is the same.
- **Workflow 1.3** [7] uses graph-based spatial clustering to aggregate nearby places with strong flows into regions, also with iterative parameter adjustment guided by a quality heatmap. *Relevant aspect*: the idea of using a quality metric display (heatmap over parameter combinations) to guide the choice of spatial aggregation parameters.

A.3 Phase 1: Aggregating Trip Data and Computing Availability

A.3.1 What You Do

With places defined, you now transform the raw trip records into place-level time series. For each place and each time interval (e.g., one hour):

- Count the number of **bikes taken** (trip origins at member stations).
- Count the number of **bikes returned** (trip destinations at member stations).
- Compute the **net flow** (returns minus takes).
- Compute **directed flows between place pairs** (how many bikes moved from place A to place B).

From the net flow, compute a **running estimate of bike availability** at each place over time and the **occupancy rate** (fraction of capacity occupied). This requires either knowing the initial bike distribution or estimating it from the data and domain knowledge.

The result is a set of time series per place: takes, returns, net flow, availability, and occupancy rate.

A.3.2 Where to Find Relevant Examples

- **Workflow 1.11** [2] begins with precisely this step: transforming raw spatio-temporal records into spatial time series by dividing territory into spatial compartments and aggregating attribute values by location and time interval. *Relevant aspect*: the spatio-temporal aggregation step producing one time series per spatial unit.
- **Workflow 1.6** [1] includes a spatio-temporal aggregation step after place delineation, where events and trajectories are aggregated by places and time intervals, producing time series of counts and statistics per place, as well as directed flows between place pairs. *Relevant aspects*: aggregating both local statistics (counts per place) and relational statistics (flows between places)—exactly what you need for understanding bike redistribution patterns.

A.4 Phase 2: Discovering Temporal Patterns through Day Clustering

A.4.1 What You Do

This is the core pattern-discovery phase. The idea is to **treat each day as an analytical unit**, characterise it by its hourly demand-supply profile across all places, then **cluster days with similar profiles** and visualise the clusters on a **calendar** to reveal weekly, seasonal, and holiday patterns.

Step 2a—Partition into daily episodes. Cut all place-level time series into daily segments. Each day is now represented by a matrix of hourly values (places \times hours) for takes, returns, net flow, and availability.

Step 2b—Compute daily profiles. Summarise each day by a feature vector that captures the system-wide hourly shape of demand and supply—for instance, the total takes and returns per hour across all places, plus indicators like total volume and peak-hour timing.

Step 2c—Cluster days iteratively. Apply hierarchical clustering (or another method) to group days with similar profiles. Adjust the number of clusters and distance measure interactively until you get a clear, interpretable decomposition.

Step 2d—Visualise and interpret. Use three coordinated views:

1. **Calendar view:** A grid where each cell is one day, coloured by cluster membership. Months run along one axis, days of the week along the other. This immediately reveals whether clusters correspond to weekdays vs. weekends, holidays, seasons, etc.
2. **Profile line graphs:** For each cluster, show the average hourly take and return curves (with variability bands). These show the characteristic diurnal shape for each day type—e.g., “regular weekday” might show morning peaks at residential places and evening peaks at business areas.
3. **Place-level heatmap:** A matrix (places as rows, hours as columns, one panel per cluster) where colour intensity shows the net flow at each place and hour. Blue might indicate places gaining bikes, red places losing bikes. This reveals *where* demand imbalances occur for each day type.

Optionally, also show **flow map thumbnails** for selected hours (e.g., morning peak, evening peak) with arrows between places sized by flow volume, to visualise the dominant movement patterns.

Examine these views together. Assign interpretive labels to clusters (e.g., “regular weekday,” “summer weekend,” “public holiday”). If clusters are not yet clear or too many/too few, adjust parameters and re-cluster.

A.4.2 Where to Find Relevant Examples

- **Workflow 1.1** [6] is the **primary reference** for this phase. It defines exactly this approach: partition time series into daily episodes, characterise each day by its temporal profile, hierarchically cluster days by profile similarity, and display results through a calendar view (colour-coded by cluster) coordinated with line graphs of cluster-average profiles. The analyst iteratively adjusts the number of clusters, distance measure, and time interval focus until meaningful patterns emerge. *Relevant aspects:* the entire iterative loop of clustering → calendar + profile visualisation → interpretation → assessment → parameter adjustment. Your workflow extends this by making the daily profiles multi-place (matrices instead of single vectors) and adding the place-level heatmap, but the core logic is directly from this paper.
- **Workflow 1.3** [7] applies temporal clustering to flow data (grouping time steps with similar spatial flow patterns) and displays results on a calendar with flow graph thumbnails per cluster. *Relevant aspects:* the calendar view for temporal cluster distribution combined with small-multiple flow graph thumbnails showing representative spatial patterns per cluster—a visualisation design directly applicable to your flow map thumbnails.
- **Workflow 1.9** [3] provides a model for progressive abstraction of multivariate temporal data. It shows how symbolic encoding of temporal patterns within episodes, followed by topic modelling to discover co-occurring patterns, can reveal multi-attribute behaviours. *Relevant aspects:* the general strategy of encoding temporal variation within episodes and then discovering higher-level patterns from the encoded representations, especially useful if you want to go beyond simple day clustering and discover more nuanced combinations of place-level patterns.

A.5 Phase 3: Identifying and Analysing Critical Events

A.5.1 What You Do

Define **critical events** as time intervals when a place’s occupancy rate crosses a threshold—either too low (shortage: few bikes available, users cannot take a bike) or too high (overcrowding: few

free docks, users cannot return a bike). Set thresholds based on domain knowledge (e.g., below 10% of capacity = shortage, above 90% = overcrowding).

Scan the availability time series to **extract all critical episodes**: for each, record the place, type (shortage or overcrowding), start and end time, duration, and severity (e.g., estimated number of unserved users).

Then **aggregate** critical events by place, day type (from Phase 2), and hour of day to build a profile of *where* and *when* shortages and overcrowding concentrate.

Visualise the results with:

- A **map** where each place has a glyph sized by total critical event frequency, coloured by type (red for shortage, blue for overcrowding), with an embedded hourly bar diagram showing at which hours critical events occur.
- A **calendar heatmap** where cell colour intensity reflects the number of critical events per day.

Interpret these views together with the Phase 2 results. Look for **spatial complementarity**—e.g., morning shortages at residential places co-occurring with overcrowding at business-area places. These complementary patterns are the key to designing effective relocation strategies.

A.5.2 Where to Find Relevant Examples

- **Workflow 1.6** [1] provides the model for **event extraction and characterisation**: identifying relevant events from data using attribute-based criteria, then aggregating and visualising events by places and time intervals to discover spatio-temporal patterns. *Relevant aspects*: the event extraction step (applying threshold conditions to identify events), spatio-temporal aggregation of events per place and time interval, and exploration through temporal diagrams positioned on a map.
- **Workflow 1.1** [6] and **Workflow 1.3** [7]—the calendar-based visualisation approach from Phase 2 can be reused here to show the temporal distribution of critical events across the calendar. *Relevant aspect*: the calendar grid as a tool for revealing weekly and seasonal concentration patterns.

A.6 Phase 4: Building Predictive Demand-Supply Models

A.6.1 What You Do

The goal is to build a model that, given any date and time of day, predicts the expected takes, returns, and net flow at each place. This model is the foundation for the allocation strategy in Phase 5.

Step 4a—Group places by demand similarity. Cluster places whose demand-supply time series have similar temporal shapes (even if different magnitudes). This allows you to fit a shared model structure per group, with per-place scaling parameters, rather than building a separate model for each place.

Step 4b—Identify temporal components. For each place group, use the cluster profiles from Phase 2 to identify the temporal variation components: the diurnal cycle shape, weekly modulation (weekday vs. weekend), seasonal variation, and holiday effects.

Step 4c—Fit a time series model. Derive a representative time series for each place group. Select and configure a modelling method that can handle multiple seasonal components (e.g., a multiplicative seasonal model with 24-hour and 168-hour cycles, plus seasonal and holiday adjustment factors). Fit the model and overlay the model curve on the actual data.

Step 4d—Iteratively refine. Assess the model visually: does the model curve capture the characteristic diurnal shape? The weekday/weekend distinction? The seasonal variation? If not, adjust parameters and refit.

Step 4e—Evaluate through residual analysis. Compute residuals (actual minus predicted) for all places and examine their distribution over time and space. Residuals should look random; if they show systematic patterns (e.g., consistently underestimating demand at certain hours or places), this indicates the model needs refinement—perhaps additional components, different grouping, or special handling of holidays.

This is a **nested loop**: an outer loop over the grouping–modelling–residual cycle, and inner loops for cluster refinement and model parameter tuning.

A.6.2 Where to Find Relevant Examples

- **Workflow 1.11** [2] is the **primary reference** for the entire modelling phase. It describes precisely this approach: cluster spatial time series by temporal similarity, visually identify temporal variation characteristics, derive representative series, configure and fit statistical time series models, iteratively refine model parameters while comparing model curves to data, and evaluate model quality by examining residual distributions over time and space. If residuals show systematic patterns, the analyst decides whether to subdivide clusters, adjust the modelling approach, or both. *Relevant aspects*: essentially the entire modelling workflow—grouping → representative derivation → model configuration → fitting → visual comparison → residual-based evaluation → refinement decision. Your workflow adds holiday/seasonal exogenous variables and multi-scale seasonality, but the structural logic comes directly from this paper.
- **Workflow 1.10** [5] provides a complementary approach to model refinement through residual analysis. After building an initial model, residuals become the analytical target: features are re-ranked by their relevance to the residuals, revealing effects not yet captured by the model, and the analyst decides which features or interactions to add. *Relevant aspect*: the residual-based discovery loop—compute residuals → re-rank features by residual relevance → visualise conditional residual distributions → discover unexplained effects → refine model. This approach is useful if you want a more structured way to decide what to add to your model when residuals are non-random.

A.7 Phase 5: Developing and Validating the Allocation Strategy

A.7.1 What You Do

With a predictive model in hand, you now build an **allocation/relocation model**: given a date and time, use the demand-supply predictions to compute a recommended distribution of bikes across places that minimises expected shortages and overcrowding.

Step 5a—Define objectives and constraints. Specify what you are optimising: minimise total expected shortage and overcrowding events, subject to constraints—total fleet size, place capacities, and logistical limits (e.g., maximum bikes a redistribution vehicle can move per trip, operating hours of redistribution crews).

Step 5b—Formulate the optimisation model. Build a model that takes the predicted demand curves for a given date and time, and computes the optimal initial bike allocation and (optionally) a schedule of relocations during the day.

Step 5c—Validate by simulation. Test the model on historical dates: for a representative sample of dates (sampling from each day-type cluster identified in Phase 2), compute the recommended allocation, simulate the resulting availability through the day using actual demand data, and count how many critical events (shortages and overcrowding) would have been avoided compared to the actual historical situation.

Step 5d—Visualise and assess. Display a side-by-side comparison: for each place, show the critical event frequency under the actual historical allocation versus the recommended allocation. Use a map with paired bars (actual vs. recommended) and a summary table by day type showing the total reduction in critical events. Assess whether the improvement is sufficient. If certain places or day types still show unacceptable critical event rates, refine the allocation criteria—for example, adjust the relative weighting of shortage vs. overcrowding, introduce priority places, or add time-of-day relocation windows.

This is again an **iterative loop**: specify criteria → build allocation model → simulate on historical data → visualise comparison → assess → refine criteria → rebuild.

A.7.2 Where to Find Relevant Examples

- **Workflow 1.5** [4] provides the model for the **prescriptive recommendation loop**. In that paper, the analyst specifies an action plan, the system estimates the plan’s impact by recomputing outcome probabilities, the analyst assesses the result, and if unsatisfied, refines the plan based on outcome feedback. This cycle continues until the analyst is satisfied. *Relevant aspects*: the iterative plan-specification → impact-estimation → visualisation → assessment → refinement cycle. In your case, the “action plan” is the bike allocation, the “outcome estimation” is the simulated critical event count, and the “refinement” is adjusting allocation criteria. The structural logic is the same.

A.8 Phase 6: Synthesising Knowledge

A.8.1 What You Do

At the end, bring together all findings into a comprehensive understanding:

1. **Demand-supply pattern types**: The named day types (from Phase 2) with their characteristic diurnal shapes, place-level variations, and calendar distribution.
2. **Critical event patterns**: Which places are most affected, when, and why—including spatial complementarity patterns (paired shortages and overcrowding) explained by directional flows.
3. **Predictive models**: For any given date and time, the expected demand and supply at each place, with the model’s accuracy and limitations documented.
4. **Allocation recommendations**: The validated relocation strategy with its expected reduction in critical events, along with operational guidance for redistribution logistics.

All 17 workflows in the library end with a knowledge synthesis step. This case combines pattern-based understanding (as in Workflows 1.1, 1.3, 1.6) with model-based prediction (as in Workflow 1.11) and prescriptive recommendation (as in Workflow 1.5).

A.9 Summary of Key References

Table 2 summarises, for each phase, the primary reference workflows and what to look for in the corresponding papers.

Table 2: Primary references for each workflow phase.

Phase	What You Do	Primary Reference	What to Look at in That Paper
0	Group stations into places	1.6 [1]	Iterative spatial clustering with visual assessment on map
1	Aggregate to place-level time series	1.11 [2], 1.6 [1]	Spatio-temporal aggregation step; aggregation of local counts and inter-place flows
2	Cluster days; calendar + profiles	1.1 [6], 1.3 [7]	Day clustering + calendar view + profile line graphs (1.1); temporal clustering + calendar + flow thumbnails (1.3)
3	Identify critical events	1.6 [?]	Event extraction by threshold conditions; spatio-temporal aggregation of events; temporal diagrams on map
4	Build predictive model	1.11 [2], 1.10 [5]	Full modelling workflow: grouping → representative → model fitting → residual evaluation (1.11); residual-based feature discovery loop (1.10)
5	Allocation / relocation strategy	1.5 [4]	Iterative plan specification → impact estimation → assessment → refinement loop

A.10 ATWL representation

```

workflow bike-sharing-spatio-temporal-analysis

template:
  loop(define-unit (places) → visualise → assess) →
  characterise (aggregate) → characterise (availability) →
  define-unit (daily episodes) → characterise (profiles) →
  loop(define-unit (cluster days) → characterise (cluster summaries) →
    contextualise (calendar) → visualise → abstract → assess) →
  define-unit (critical events) → visualise → abstract →
  define-unit (group places) →
  loop(abstract (cycles) → generate-knowledge (configure) →
    build-model → visualise → assess →
    characterise (residuals) → visualise → assess) →
  build-model (allocation) →
  loop(characterise (simulate) → visualise → assess) →
  generate-knowledge

description: "Analyse spatio-temporal patterns of public bike usage at
  the level of places (groups of nearby docking stations), reveal
  patterns of bike shortage and station overcrowding related to
  diurnal, weekly, seasonal cycles and holidays, build predictive
  demand-supply models, and develop a bike allocation/relocation
  strategy"

# =====
# GIVEN ARTIFACTS
# =====

artifact D_trips : entities

```

```
origin: given
internal structure: elementary
embedment: {set, time}
features:
  - id: bike_id
    value structure: atomic
    value type: reference
    description: "Bike identifier"
  - id: origin_station
    value structure: atomic
    value type: reference
    description: "Origin docking station"
  - id: origin_time
    value structure: atomic
    value type: temporal
    description: "Trip start time"
  - id: dest_station
    value structure: atomic
    value type: reference
    description: "Destination docking station"
  - id: dest_time
    value structure: atomic
    value type: temporal
    description: "Trip end time"
description: "Individual bike trip records with origin and
destination stations and timestamps"
```

artifact D_stations : entities

```
origin: given
internal structure: elementary
embedment: space
features:
  - id: coordinates
    value structure: atomic
    value type: spatial
    description: "Geographic location"
  - id: capacity
    value structure: atomic
    value type: numeric
    description: "Number of docking positions"
description: "Docking stations with geographic locations and
docking capacities"
```

artifact D_calendar : entities

```
origin: given
internal structure: elementary
embedment: time
features:
  - id: day_of_week
    value structure: atomic
    value type: categorical
    description: "Day of week (Mon-Sun)"
  - id: month
    value structure: atomic
    value type: categorical
    description: "Month"
  - id: season
    value structure: atomic
    value type: categorical
    description: "Season"
  - id: is_holiday
    value structure: atomic
    value type: categorical
```

```

    description: "Holiday indicator (yes/no)"
  - id: day_type
    value structure: atomic
    value type: categorical
    description: "Working day, weekend, or public holiday"
description: "Calendar structure with day-of-week, month,
season, and holiday annotations"

artifact K_domain : knowledge(D_trips, D_stations)
  origin: given
  representation form: "domain constraints and assumptions"
  description: "Domain knowledge: system operating hours, fleet
size, initial bike distribution (or estimation method),
known relocation operations, special event calendar"

# =====
# PHASE 0: PLACE DEFINITION
# Adapted from 1.6 (place delineation) and 1.3 (spatial aggregation),
# simplified because station locations are given.
# =====

artifact S_station_clustering : specification
  origin: given
  representation form: "parameter settings"
  description: "Spatial clustering parameters: distance threshold
for grouping nearby stations, minimum group size"

loop L_place_definition:
  purpose: "Group spatially close docking stations into places
that serve as analytical units for the entire workflow"
  until: "Places are spatially coherent, popular areas with
multiple stations are properly grouped, and place
granularity is appropriate for pattern analysis"
  body:

    transform T_cluster_stations :
      intent: define-unit
      manner: "spatial clustering by proximity"
      input: D_stations, S_station_clustering
      output: D_places, F_place_membership
      actor: machine
      description: "Group spatially close stations into places;
compute aggregate capacity per place"

    artifact D_places : entities
      internal structure: group/cluster
      embedment: space
      features:
        - id: centroid
          value structure: atomic
          value type: spatial
          description: "Centroid location of member stations"
        - id: total_capacity
          value structure: atomic
          value type: numeric
          description: "Sum of docking capacities of members"
        - id: num_stations
          value structure: atomic
          value type: numeric
          description: "Number of member stations"
      description: "Groups of nearby docking stations forming
place-level analytical units with aggregate capacities"

```

```

artifact F_place_membership : feature(D_stations)
  value structure: atomic
  value type: categorical
  description: "Place assignment for each station"

transform T_vis_places :
  intent: visualise
  manner: "stations and places on geographic map"
  input: D_stations, D_places, F_place_membership
  output: V_places_map
  actor: machine
  description: "Display stations colored by place membership
    on map with place boundaries and capacity indicators"

artifact V_places_map : visualisation(D_stations, D_places,
  F_place_membership)
  layout: "geographic map"
  form: "colored points with cluster boundaries"
  encoding: "station position from coordinates; color from
    place membership; boundary from convex hull or buffer;
    place label size proportional to total capacity"
  description: "Map showing station grouping into places"

transform T_assess_places :
  intent: assess
  manner: "evaluate spatial coherence and granularity"
  input: V_places_map, D_places, S_station_clustering
  output: K_place_assessment
  actor: human
  description: "Assess whether places are spatially coherent,
    popular areas are properly grouped, and granularity
    is appropriate"

artifact K_place_assessment : knowledge(D_places)
  representation form: "quality judgment"
  description: "Assessment of place quality and whether
    clustering parameter adjustment is needed"

if K_place_assessment indicates adjustment needed:
  then:
    transform T_adjust_station_clustering :
      intent: generate-knowledge
      manner: "adjust spatial clustering parameters"
      input: K_place_assessment, V_places_map,
        S_station_clustering
      output: S_station_clustering'
      actor: human
      description: "Adjust distance threshold or minimum size"

    artifact S_station_clustering' : specification
      representation form: "parameter settings"
      description: "Updated clustering parameters"

    assign:
      S_station_clustering := S_station_clustering'
  else:
    exit loop L_place_definition

end loop L_place_definition

# =====

```

```

# PHASE 1: SPATIO-TEMPORAL AGGREGATION
# Adapted from 1.11 (spatio-temporal aggregation) and 1.6 (place-level
# aggregation with flows).
# =====

artifact S_time_granularity : specification
  origin: given
  representation form: "parameter settings"
  description: "Time interval for aggregation (e.g., 1 hour)"

transform T_aggregate_trips :
  intent: characterise
  manner: "spatio-temporal aggregation by places and time intervals"
  input: D_trips, D_places, F_place_membership, S_time_granularity
  output: F_take_series, F_return_series, F_flow_series
  actor: machine
  description: "Count bike takes and returns per place per time
  interval; compute directed flows between place pairs over time"

artifact F_take_series : feature(D_places)
  value structure: vector
  value type: numeric
  representation form: "time series per place"
  description: "Time series of bike take counts per place
  per time interval"

artifact F_return_series : feature(D_places)
  value structure: vector
  value type: numeric
  representation form: "time series per place"
  description: "Time series of bike return counts per place
  per time interval"

artifact F_flow_series : feature(D_places)
  value structure: relational configuration
  value type: {reference, numeric}
  representation form: "directed flows with time series"
  description: "Time series of directed bike flow counts between
  each pair of places"

transform T_compute_availability :
  intent: characterise
  manner: "compute running availability from cumulative net flow"
  input: D_places, F_take_series, F_return_series, K_domain
  output: F_net_flow, F_availability, F_occupancy_rate
  actor: machine
  description: "Compute net flow (returns minus takes), running
  bike availability, and occupancy rate as fraction of
  capacity, using initial distribution from domain knowledge"

artifact F_net_flow : feature(D_places)
  value structure: vector
  value type: numeric
  representation form: "time series per place"
  description: "Time series of net bike flow (returns - takes)
  per place per interval"

artifact F_availability : feature(D_places)
  value structure: vector
  value type: numeric
  representation form: "time series per place"
  description: "Estimated available bike count per place over
  time, derived from cumulative net flow and initial state"

```

```

artifact F_occupancy_rate : feature(D_places)
  value structure: vector
  value type: numeric
  representation form: "time series per place"
  description: "Fraction of capacity occupied at each place
    over time (0 = empty, 1 = full)"

# =====
# PHASE 2: TEMPORAL PATTERN DISCOVERY
# Adapted from 1.1 (cluster-calendar) extended with multi-place profiles,
# and from 1.3 (temporal clustering with calendar view and flow thumbnails).
# =====

transform T_partition_days :
  intent: define-unit
  manner: "time-partitioning into daily episodes"
  input: D_places, F_take_series, F_return_series, F_net_flow,
    F_availability
  output: D_days
  actor: machine
  description: "Partition all place-level time series into daily
    episodes, each containing hourly profiles for all places"

artifact D_days : entities
  internal structure: episode
  embedment: time
  features:
    - id: date
      value structure: atomic
      value type: temporal
      description: "Calendar date"
    - id: take_profiles
      value structure: matrix
      value type: numeric
      description: "Hourly take counts: places x hours"
    - id: return_profiles
      value structure: matrix
      value type: numeric
      description: "Hourly return counts: places x hours"
    - id: net_flow_profiles
      value structure: matrix
      value type: numeric
      description: "Hourly net flow: places x hours"
    - id: availability_profiles
      value structure: matrix
      value type: numeric
      description: "Hourly availability: places x hours"
  description: "Daily episodes containing hourly demand-supply
    profiles across all places"

transform T_profile_days :
  intent: characterise
  manner: "compute composite daily profiles"
  input: D_days
  output: F_day_profile
  actor: machine
  description: "Represent each day by a feature vector capturing
    system-wide hourly take and return patterns (aggregated
    across places) and, optionally, place-level patterns"

artifact F_day_profile : feature(D_days)

```

```

value structure: vector
value type: numeric
description: "Composite daily profile: system-wide hourly
take and return curves, total volume, and peak-hour
indicators"

artifact S_day_clustering : specification
origin: given
representation form: "parameter settings"
description: "Day clustering parameters: number of clusters,
distance measure, features included"

loop L_day_clustering:
purpose: "Cluster days with similar demand-supply patterns
to reveal diurnal, weekly, seasonal, and holiday
regularities"
until: "Day clusters are interpretable, well-separated, and
reveal clear correspondence with weekday/weekend/holiday
and seasonal structure"
body:

transform T_cluster_days :
intent: define-unit
manner: "hierarchical clustering by profile similarity"
input: D_days, F_day_profile, S_day_clustering
output: D_day_clusters, F_day_cluster_label
actor: hybrid
description: "Group days with similar system-wide
demand-supply profiles"

artifact D_day_clusters : entities
internal structure: group/cluster
embedment: set
features:
- id: cluster_size
value structure: atomic
value type: numeric
description: "Number of days in cluster"
description: "Groups of days with similar demand-supply
patterns"

artifact F_day_cluster_label : feature(D_days)
value structure: atomic
value type: categorical
description: "Cluster membership for each day"

transform T_aggregate_clusters :
intent: characterise
manner: "compute cluster-average profiles and flow summaries"
input: D_day_clusters, F_day_profile, D_days,
F_take_series, F_return_series, F_flow_series
output: F_cluster_profile, F_cluster_flow
actor: machine
description: "Compute average hourly demand-supply profiles
per cluster at system and place level; compute average
inter-place flow patterns per cluster"

artifact F_cluster_profile : feature(D_day_clusters)
value structure: vector
value type: numeric
description: "Cluster-average hourly profiles of takes,
returns, and net flow, both system-wide and per place"

```

```

artifact F_cluster_flow : feature(D_day_clusters)
  value structure: relational configuration
  value type: {reference, numeric}
  representation form: "average directed flows per cluster
    for selected hours (e.g., morning peak, evening peak)"
  description: "Cluster-average inter-place flow patterns"

transform T_arrange_calendar :
  intent: contextualise
  manner: "calendar-based"
  input: D_days, D_calendar
  output: A_calendar
  actor: machine
  description: "Arrange daily episodes in calendar context"

artifact A_calendar : arrangement(D_days)
  context: D_calendar
  principle: "calendar date mapping to grid position
    (weeks as rows, days of week as columns)"
  description: "Calendar arrangement positioning each day
    within its weekly and monthly context"

# Coordinated views adapted from 1.1 (calendar + profiles)
# and 1.3 (calendar + thumbnails + flow maps)

transform T_vis_calendar :
  intent: visualise
  manner: "calendar grid with color-coded clusters"
  input: A_calendar, F_day_cluster_label
  output: V_calendar
  actor: machine
  description: "Display days on calendar colored by cluster
    membership, highlighting holiday annotations"

artifact V_calendar : visualisation(A_calendar, F_day_cluster_label)
  layout: "calendar grid (months as row groups, weekdays
    as columns)"
  form: "colored cells with holiday markers"
  encoding: "position from A_calendar; color from cluster
    label; markers for holidays"
  description: "Calendar view revealing weekly and seasonal
    distribution of demand-supply pattern types"

transform T_vis_profiles :
  intent: visualise
  manner: "line graphs and flow thumbnails per cluster"
  input: F_cluster_profile, F_cluster_flow, D_day_clusters,
    D_places
  output: V_profiles
  actor: machine
  description: "Display cluster-average daily take/return
    profiles and representative flow patterns"

artifact V_profiles : visualisation(F_cluster_profile,
  F_cluster_flow, D_day_clusters, D_places)
  layout: "one panel per cluster: line graphs above,
    flow map thumbnails below (morning/evening peaks)"
  form: "line graphs with confidence bands; directed
    flow arrows on small maps"
  encoding: "x: hour of day; y: average count; separate
    lines for takes (outgoing) and returns (incoming);
    flow arrows: width from magnitude, color from
    cluster; bands for within-cluster variability"

```

```

description: "Characteristic diurnal demand-supply
  curves and spatial flow patterns per day type"

transform T_vis_place_heatmap :
  intent: visualise
  manner: "heatmap of place-level demand variation"
  input: F_cluster_profile, D_places, D_day_clusters
  output: V_place_heatmap
  actor: machine
  description: "Display place-level hourly demand patterns
    per cluster as heatmap"

artifact V_place_heatmap : visualisation(F_cluster_profile,
  D_places, D_day_clusters)
  layout: "matrix: places as rows (ordered by geography
    or cluster), hours as columns, one panel per
    day-type cluster"
  form: "colored cells"
  encoding: "color intensity from net flow value; blue for
    net bike gain, red for net bike loss"
  description: "Heatmap revealing which places gain or lose
    bikes at which hours for each day type"

transform T_interpret_patterns :
  intent: abstract
  manner: "identify temporal pattern types and associations"
  input: V_calendar, V_profiles, V_place_heatmap,
    D_day_clusters
  output: P_temporal_patterns
  actor: human
  description: "Interpret cluster meanings: associate with
    weekdays, weekends, holidays, seasons; note
    characteristic diurnal shapes and place-level
    variations"

artifact P_temporal_patterns : pattern(D_day_clusters, D_places)
  representation form: "labeled pattern types with descriptions"
  description: "Named demand-supply pattern types (e.g.,
    'regular weekday', 'weekend', 'summer weekday',
    'public holiday') with diurnal shape descriptions
    and place-specific variations"

transform T_assess_day_clusters :
  intent: assess
  manner: "evaluate cluster quality and interpretability"
  input: V_calendar, V_profiles, V_place_heatmap,
    P_temporal_patterns, D_day_clusters
  output: K_cluster_assessment
  actor: human
  description: "Assess whether clusters provide clear
    decomposition aligned with calendar structure"

artifact K_cluster_assessment : knowledge(D_day_clusters)
  representation form: "quality judgment"
  description: "Assessment of cluster quality: separation,
    interpretability, calendar correspondence, and
    whether refinement is needed"

if K_cluster_assessment indicates refinement needed:
  then:
    transform T_adjust_day_clustering :
      intent: generate-knowledge
      manner: "adjust clustering parameters"

```

```

    input: K_cluster_assessment, V_calendar, V_profiles,
           S_day_clustering
    output: S_day_clustering'
    actor: human
    description: "Adjust number of clusters, distance
                measure, or feature composition"

    artifact S_day_clustering' : specification
        representation form: "parameter settings"
        description: "Updated day clustering parameters"

    assign:
        S_day_clustering := S_day_clustering'
    else:
        exit loop L_day_clustering

end loop L_day_clustering

# =====
# PHASE 3: CRITICAL EVENT ANALYSIS
# Adapted from 1.6 (event extraction and characterisation) with
# new critical-state definitions.
# =====

artifact S_critical_thresholds : specification
    origin: given
    representation form: "parameter settings"
    description: "Thresholds for critical states: minimum available
                bikes for shortage (e.g., <10% capacity), minimum free
                docks for overcrowding (e.g., <10% remaining capacity)"

transform T_identify_critical :
    intent: define-unit
    manner: "extract episodes where availability crosses thresholds"
    input: D_places, F_availability, F_occupancy_rate,
           S_critical_thresholds
    output: D_critical_events
    actor: machine
    description: "Identify time intervals and places where bike
                shortage or dock overcrowding occurs"

artifact D_critical_events : entities
    internal structure: episode
    embedment: {space, time}
    features:
        - id: place_ref
          value structure: atomic
          value type: reference
          description: "Place where critical event occurs"
        - id: event_type
          value structure: atomic
          value type: categorical
          description: "'shortage' or 'overcrowding'"
        - id: duration
          value structure: atomic
          value type: temporal
          description: "Duration of critical state"
        - id: severity
          value structure: atomic
          value type: numeric
          description: "Depth of critical state (e.g., number of
                unserved take or return attempts estimated from

```

```

    demand during shortage)"
description: "Episodes of bike shortage or dock overcrowding
    at specific places and times"

transform T_characterise_critical :
    intent: characterise
    manner: "aggregate critical events by place, day type, and
        time of day"
    input: D_critical_events, D_places, F_day_cluster_label, D_days
    output: F_critical_frequency
    actor: machine
    description: "Compute frequency, duration, and severity of
        critical events per place, per day type, and per hour"

artifact F_critical_frequency : feature(D_places)
    value structure: matrix
    value type: numeric
    representation form: "critical event frequency matrix
        (place x hour x day-type)"
    description: "Frequency and severity of shortage and
        overcrowding by place, hour, and day type"

transform T_vis_critical :
    intent: visualise
    manner: "geographic map with temporal diagrams and calendar
        heatmap"
    input: D_places, F_critical_frequency, A_calendar,
        F_day_cluster_label, D_critical_events
    output: V_critical
    actor: machine
    description: "Display critical event patterns on map (where)
        and calendar (when)"

artifact V_critical : visualisation(D_places, F_critical_frequency,
    A_calendar)
    layout: "geographic map with embedded hourly bar diagrams;
        calendar heatmap panel"
    form: "sized bicolored glyphs on map with bar diagrams;
        colored calendar cells"
    encoding: "map: glyph position from place centroid; glyph
        size from total critical frequency; red for shortage,
        blue for overcrowding; bar diagram shows hourly profile
        of critical events. Calendar: cell color intensity from
        critical event count per day"
    description: "Coordinated spatial and temporal views of
        where and when shortages and overcrowding occur"

transform T_interpret_critical :
    intent: abstract
    manner: "identify critical spatio-temporal patterns"
    input: V_critical, P_temporal_patterns, F_critical_frequency,
        V_place_heatmap
    output: P_critical_patterns
    actor: human
    description: "Identify which places are most affected, when
        critical events concentrate, how they relate to day
        types and flows, and whether spatial complementarity
        exists (shortage at place A co-occurring with
        overcrowding at nearby place B)"

artifact P_critical_patterns : pattern(D_places, D_critical_events)
    representation form: "categorised critical patterns with
        descriptions"

```

```

description: "Identified critical patterns: places prone to
shortage vs. overcrowding, time-of-day and day-type
concentrations, spatial complementarity (e.g., morning
shortage in residential areas paired with overcrowding
in business areas), seasonal variation of critical
events"

# =====
# PHASE 4: PREDICTIVE MODEL BUILDING
# Adapted from 1.11 (time series clustering, representative derivation,
# model fitting, and residual-based evaluation). Extended with holiday
# and seasonal exogenous variables.
# =====

# Group places by temporal demand similarity for joint modelling

transform T_group_places :
  intent: define-unit
  manner: "cluster by temporal demand-supply similarity"
  input: D_places, F_take_series, F_return_series
  output: D_place_groups, F_place_group_label
  actor: hybrid
  description: "Group places with similar temporal demand-supply
patterns so that a shared predictive model can be fitted
per group, with per-place scaling parameters"

artifact D_place_groups : entities
  internal structure: group/cluster
  embedment: set
  description: "Groups of places with similar temporal
demand-supply patterns"

artifact F_place_group_label : feature(D_places)
  value structure: atomic
  value type: categorical
  description: "Place group membership"

artifact S_model_config : specification
  origin: given
  representation form: "model configuration"
  description: "Initial model configuration: model type
(e.g., multiplicative seasonal model), cycle lengths
(diurnal=24h, weekly=168h), seasonal component, holiday
effect handling"

artifact K_modelling_strategy : knowledge
  origin: given
  representation form: "refinement strategy"
  description: "Modelling guidance from prior residual analysis;
initially neutral"

loop L_modelling:
  purpose: "Build and refine predictive models of bike demand
and supply per place group until models capture diurnal,
weekly, seasonal, and holiday effects"
  until: "Model residuals show no systematic temporal or spatial
patterns; predictions capture demand-supply dynamics
across all day types"
  body:

    # Identify temporal characteristics (from 1.11 Step 2a)
    transform T_identify_cycles :

```

```

intent: abstract
manner: "identify periodic and trend components"
input: V_profiles, P_temporal_patterns, D_place_groups
output: P_temporal_chars
actor: human
description: "For each place group, identify diurnal cycle
  shape, weekly modulation, seasonal variation,
  and holiday effects from cluster profiles"

artifact P_temporal_chars : pattern(D_place_groups)
representation form: "identified temporal characteristics"
description: "Temporal variation characteristics per
  place group: cycle lengths, seasonal amplitude,
  holiday effect type and magnitude"

# Configure model (from 1.11 Step 2c)
transform T_configure_model :
  intent: generate-knowledge
  manner: "specify model structure from identified cycles"
  input: P_temporal_chars, K_modelling_strategy,
    S_model_config
  output: S_model_config'
  actor: human
  description: "Configure model: select method, set cycle
    parameters, specify holiday and seasonal handling"

artifact S_model_config' : specification
representation form: "model configuration"
description: "Specified model structure with cycle lengths,
  seasonal components, holiday indicators, and method"

assign:
  S_model_config := S_model_config'

# Derive representative series (from 1.11 Step 2b)
transform T_derive_representative :
  intent: characterise
  manner: "compute group-representative demand-supply series"
  input: D_place_groups, F_take_series, F_return_series
  output: F_representative
  actor: machine
  description: "Derive representative take and return time
    series for each place group from member statistics"

artifact F_representative : feature(D_place_groups)
value structure: vector
value type: numeric
description: "Representative demand-supply time series
  per place group"

# Build model (from 1.11 Step 2c/d)
transform T_build_demand_model :
  intent: build-model
  manner: "fit time series model with multiple seasonal
    components and holiday effects"
  input: F_representative, S_model_config, D_calendar
  output: M_demand_supply
  actor: machine
  description: "For each place group, fit a time series model
    capturing diurnal cycle, weekly modulation, seasonal
    variation, and holiday adjustments"

artifact M_demand_supply : model(F_representative, S_model_config)

```

```

model type: "seasonal time series model"
representation form: "fitted parametric model with
  multiple seasonal components and calendar effects"
description: "Predictive model of bike demand (takes) and
  supply (returns) per place group, incorporating
  diurnal, weekly, and seasonal cycles with holiday
  adjustment factors"

# Visualise model fit (from 1.11 Step 2d)
transform T_vis_model :
  intent: visualise
  manner: "model curves overlaid on data"
  input: M_demand_supply, F_representative, F_take_series,
    F_return_series
  output: V_model
  actor: machine
  description: "Display model predictions overlaid on actual
    data for each place group across different day types"

artifact V_model : visualisation(M_demand_supply,
  F_representative)
  layout: "temporal axis, one panel per place group; optional
    panel per day type"
  form: "line graphs with confidence bands"
  encoding: "x: time; y: count; solid lines for model;
    dashed lines for data; bands for prediction interval;
    faceted by day type"
  description: "Model-vs-data comparison revealing accuracy
    across temporal cycles and day types"

# Assess fit (from 1.11 Step 2d)
transform T_assess_model :
  intent: assess
  manner: "evaluate model fit quality"
  input: V_model, M_demand_supply
  output: K_model_assessment
  actor: human
  description: "Assess whether model captures diurnal shape,
    weekly modulation, seasonal variation, and holiday
    effects"

artifact K_model_assessment : knowledge(M_demand_supply)
  representation form: "quality judgment"
  description: "Assessment of prediction accuracy and whether
    refinement is needed"

if K_model_assessment indicates refinement needed:
  then:
    # Residual analysis (from 1.11 Step 3)
    transform T_compute_residuals :
      intent: characterise
      manner: "compute prediction residuals per place"
      input: M_demand_supply, D_places, F_take_series,
        F_return_series, D_place_groups
      output: F_residuals
      actor: machine
      description: "Compute residuals between actual and
        predicted take/return values for all places"

    artifact F_residuals : feature(D_places)
      value structure: vector
      value type: numeric
      description: "Residual time series per place"

```

```

transform T_vis_residuals :
  intent: visualise
  manner: "residuals over time and space"
  input: F_residuals, D_places, D_calendar, D_place_groups
  output: V_residuals
  actor: machine
  description: "Display residual distributions over time
    and over space"

artifact V_residuals : visualisation(F_residuals, D_places)
  layout: "temporal panel with quintile bands;
    geographic map panel"
  form: "summary bands and colored areas"
  encoding: "temporal: quintile stripes of residuals by
    hour-of-week; spatial: map colored by mean
    residual magnitude per place"
  description: "Residual analysis views for detecting
    systematic temporal or spatial patterns"

transform T_assess_residuals :
  intent: assess
  manner: "evaluate residual randomness"
  input: V_residuals, D_place_groups
  output: K_residual_quality
  actor: human
  description: "Assess whether residuals are randomly
    distributed; identify systematic patterns"

artifact K_residual_quality : knowledge(M_demand_supply)
  representation form: "quality judgment"
  description: "Assessment of residual patterns and
    whether they indicate model deficiencies"

transform T_decide_refinement :
  intent: generate-knowledge
  manner: "diagnose residual patterns and formulate
    refinement strategy"
  input: K_residual_quality, V_residuals,
    K_model_assessment, S_model_config
  output: K_modelling_strategy', S_model_config_r
  actor: human
  description: "Determine refinement: adjust model
    parameters, add components, subdivide place
    groups, or add holiday/event variables"

artifact K_modelling_strategy' : knowledge(M_demand_supply)
  representation form: "refinement strategy"
  description: "Refinement decision: what to change
    in model or grouping"

artifact S_model_config_r : specification
  representation form: "model configuration"
  description: "Revised model configuration"

assign:
  K_modelling_strategy := K_modelling_strategy'
  S_model_config := S_model_config_r
else:
  exit loop L_modelling

end loop L_modelling

```

```

# =====
# PHASE 5: ALLOCATION / RELOCATION MODEL
# New phase inspired by 1.5 (EventAction prescriptive loop):
# specify allocation → simulate → assess → refine.
# =====

artifact S_allocation_criteria : specification
  origin: given
  representation form: "constraints and objectives"
  description: "Allocation objectives: minimise expected shortage
    and overcrowding; constraints: total fleet size, place
    capacities, logistical relocation limits (max bikes moved
    per period, vehicle capacity)"

transform T_build_allocation_model :
  intent: build-model
  manner: "formulate optimisation model for bike distribution"
  input: M_demand_supply, D_places, S_allocation_criteria,
    D_calendar
  output: M_allocation
  actor: hybrid
  description: "Formulate an optimisation model that, given a
    date and time, uses demand-supply predictions to compute
    a recommended bike distribution across places that
    minimises expected shortages and overcrowding subject
    to capacity and logistics constraints"

artifact M_allocation : model(M_demand_supply, D_places,
  S_allocation_criteria)
  model type: "constrained optimisation model"
  representation form: "objective function with demand predictions,
    capacity constraints, and logistics constraints"
  description: "Model computing optimal bike allocation across
    places for any given date and time, based on predicted
    demand-supply patterns and operational constraints"

loop L_allocation_validation:
  purpose: "Validate and refine the allocation model by
    simulating recommendations on historical data and
    assessing effectiveness at reducing critical events"
  until: "Allocation recommendations effectively reduce
    shortage and overcrowding across all day types and
    places to acceptable levels"
  body:

    transform T_simulate_allocation :
      intent: characterise
      manner: "simulate availability under recommended
        allocation across sample historical dates"
      input: M_allocation, M_demand_supply, D_days, D_places,
        D_calendar, F_availability
      output: F_simulated_availability, F_simulated_critical
      actor: machine
      description: "For a representative sample of dates (from
        each day-type cluster), compute recommended allocation
        and simulate resulting availability; compare predicted
        critical events with actual ones"

    artifact F_simulated_availability : feature(D_places)
      value structure: matrix
      value type: numeric
      description: "Simulated hourly availability per place

```

```

    under recommended allocation for sample dates"

artifact F_simulated_critical : feature(D_places)
  value structure: matrix
  value type: numeric
  description: "Predicted critical event counts (shortage
    and overcrowding) per place under recommended
    allocation vs. actual historical allocation"

transform T_vis_allocation :
  intent: visualise
  manner: "comparison of actual vs. recommended outcomes"
  input: F_simulated_critical, F_critical_frequency,
    D_places, D_day_clusters
  output: V_allocation
  actor: machine
  description: "Display side-by-side comparison of critical
    events under actual vs. recommended allocation"

artifact V_allocation : visualisation(F_simulated_critical,
  F_critical_frequency, D_places)
  layout: "geographic map with paired bar diagrams;
    summary table by day type"
  form: "paired bars on map; tabular summary"
  encoding: "map: position from place centroid; paired bars
    showing actual (left) vs. recommended (right) critical
    event frequency; red for shortage, blue for
    overcrowding; table: rows by day type, columns by
    total critical events, percentage reduction"
  description: "Comparison showing predicted reduction in
    critical events under recommended allocation"

transform T_assess_allocation :
  intent: assess
  manner: "evaluate allocation effectiveness"
  input: V_allocation, F_simulated_critical,
    S_allocation_criteria
  output: K_allocation_assessment
  actor: human
  description: "Assess whether recommendations sufficiently
    reduce critical events; identify remaining problem
    areas (places or day types where shortages or
    overcrowding persist)"

artifact K_allocation_assessment : knowledge(M_allocation)
  representation form: "quality judgment"
  description: "Assessment of allocation model effectiveness
    and remaining problem areas"

if K_allocation_assessment indicates refinement needed:
  then:
    transform T_refine_allocation :
      intent: generate-knowledge
      manner: "adjust allocation criteria or model"
      input: K_allocation_assessment, V_allocation,
        S_allocation_criteria, P_critical_patterns
      output: S_allocation_criteria'
      actor: human
      description: "Refine objectives, constraints, or
        approach: adjust shortage/overcrowding
        weighting, revise logistics constraints, add
        time-of-day relocation windows, or introduce
        priority places"

```

```

artifact S_allocation_criteria' : specification
  representation form: "constraints and objectives"
  description: "Revised allocation criteria"

assign:
  S_allocation_criteria := S_allocation_criteria'

transform T_rebuild_allocation :
  intent: build-model
  manner: "re-optimise with revised criteria"
  input: M_demand_supply, D_places,
          S_allocation_criteria, D_calendar
  output: M_allocation'
  actor: machine
  description: "Rebuild allocation model with revised
                criteria"

artifact M_allocation' : model(M_demand_supply,
  D_places, S_allocation_criteria)
  model type: "constrained optimisation model"
  representation form: "revised optimisation formulation"
  description: "Revised allocation model"

assign:
  M_allocation := M_allocation'
else:
  exit loop L_allocation_validation

end loop L_allocation_validation

# =====
# PHASE 6: KNOWLEDGE SYNTHESIS
# =====

transform T_synthesize :
  intent: generate-knowledge
  manner: "integrate patterns, models, and recommendations"
  input: P_temporal_patterns, P_critical_patterns,
          M_demand_supply, M_allocation,
          K_allocation_assessment, K_domain
  output: K_final
  actor: human
  description: "Synthesize comprehensive understanding of bike
                sharing dynamics and formulate operational recommendations"

artifact K_final : knowledge(M_demand_supply, M_allocation,
  P_temporal_patterns, P_critical_patterns)
  representation form: "statements, model descriptions, and
  operational recommendations"
  description: "Comprehensive findings: (1) identified demand-supply
  pattern types and their weekly, seasonal, and holiday
  distribution; (2) places and times prone to shortage or
  overcrowding with causal flow explanations; (3) predictive
  models of demand and supply per place for any date and time;
  (4) validated allocation/relocation recommendations with
  expected reduction in critical events; (5) operational
  guidance for bike redistribution logistics"

```