UNIVERSITÀ DI TRENTO

An Analysis of Travel Patterns in Barcelona Metro Using Tucker3 Decomposition

Giuseppe Nicola Liso

University of Trento

11 December 2024

Index

Introduction

Abstract

In recent years, the rapid expansion of densely populated cities has highlighted the critical need for efficient urban traffic planning. A key component of this effort is understanding the mobility patterns of residents and visitors.

In this presentation, we focus on a case study involving the Barcelona metro system. The data is modeled as a three-way tensor, where each element represents the number of passengers at the i-th station, during the j-th time interval, on the k-th day. Using the **Tucker3** decomposition method, we uncover spatial clusters, temporal patterns, and the interactions between them.

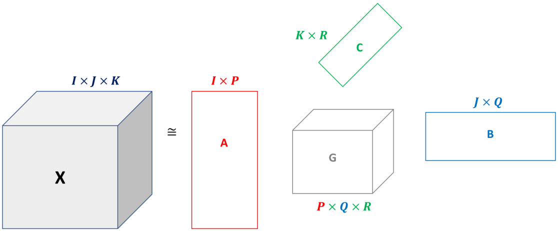
How are the data collected? Automated fare collection (AFC) systems are widely used in public transport networks. Passengers use smart cards when entering stations, recording data such as the starting and arriving station, but also the boarding and leaving time. These systems generate a large amount of data, providing useful information on urban mobility patterns. Since we can't manage big data, we need to reduce them.

Methodology

Tensors can be defined as multidimensional arrays. The **order** of a tensor, also known as **way** or **mode**, is the number of dimensions it has. Therefore: first-order tensors are **vectors**, $\mathbf{x} \in \mathbb{R}^{I_1}$, second-order tensors are **matrices**, $\mathbf{X} \in \mathbb{R}^{I_1 \times I_2}$. Tensors of order three or higher are known as higher-order tensors, $\mathbf{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$.

This work focuses on **three-way tensors**, since the data collected from smart cards can be organized in this structure.

Tucker3 decomposition is a type of tensor decomposition often used to analyze and compress multi-way data, such as three-dimensional tensors. In a three-dimensional dataset X of order $I \times J \times K$, Tucker3 decomposes the tensor into matrices along each of its three modes (directions), yielding a more compact representation that captures the essential structure of the data.



Matricization or Unfolding

To facilitate working with these big datasets, the dataset X of dimensions $I \times J \times K$ can be viewed as a stack of I frontal matrices (also called frontal slabs or slices), each of order $J \times K$. So, they can be arranged in a larger matrix called the **supermatrix** X_A. The supermatrix X_A has I rows (one for each element in the **A-mode**) and $J \times K$ columns, representing all combinations of the B-mode and C-mode elements by concatenating the frontal slices side-by-side.

This process, called **matricization** or **unfolding**, transforms the three-dimensional tensor into a two-dimensional matrix.

In Tucker3 decomposition, we aim to find component matrices for each of the three modes (A, B, and C), which helps in summarizing the data in a more compact form. Specifically, we look for:

- A: Component matrix for the A-mode (of size $I \times P$)
- B: Component matrix for the B-mode (of size $J \times Q$)
- C: Component matrix for the C-mode (of size $K \times R$)

where P, Q, and R represent the number of components chosen for each mode, which are generally less than I, J, and K, respectively.

Formalization of the Tucker3 Model

In scalar notation, the Tucker3 model can be written as follows:

$$x_{ijk} = \sum_{p=1}^{P}\sum_{q=1}^{Q}\sum_{r=1}^{R} a_{ip} b_{jq} c_{kr} g_{pqr} + e_{ijk},$$

where a_{ip}, b_{jq}, and c_{kr} denote the component scores of the i-th element on the p-th component for the **A-mode**, the j-th element on the q-th component for the **B-mode**, and the k-th element on the r-th component for the **C-mode**, respectively. The entries g_{pqr} denote the elements of the core tensor **G** and reflect the interaction among the components of the three modes, and e_{ijk} is an error term.

In matrix notation:

$$\mathbf{X}_A = \mathbf{A}\mathbf{G}_A(\mathbf{C}^T \otimes \mathbf{B}^T) + \mathbf{E}_A$$

where:

- $\mathbf{X}_A$, $\mathbf{G}_A$, and $\mathbf{E}_A$ (the error matrix) denote the $I \times JK$, $P \times QR$, and $I \times JK$ unfoldings of $\mathbf{X}$, $\mathbf{G}$, and $\mathbf{E}$, respectively.
- $\otimes$: Denotes the **Kronecker product** between matrices $\mathbf{C}^T$ and $\mathbf{B}^T$. Given two matrices $\mathbf{U}$ and $\mathbf{V}$, the Kronecker product $\mathbf{U} \otimes \mathbf{V}$ is defined as:

$$\mathbf{U} \otimes \mathbf{V} = \begin{bmatrix} u_{11}\mathbf{V} & \cdots & u_{1J}\mathbf{V} \\ \vdots & \ddots & \vdots \\ u_{I1}\mathbf{V} & \cdots & u_{IJ}\mathbf{V} \end{bmatrix}$$

Steps in Tucker3 Decomposition

1. **Unfold** X into X_A by concatenating frontal slices of X side-by-side.
2. **Factorize** X_A into a product of matrices:
   - A, B, and C capture the main structure of the data for each mode.
   - The core tensor G contains the interactions among these components.
3. **Reconstruct** X using the component matrices and the core tensor, along with any residual error term E if necessary.

Goal in Tucker3 Decomposition

The goal in Tucker3 is to find the optimal parameter matrices A, B, C, and G that best approximate the data array X while minimizing the reconstruction error. The **reconstruction error** is measured by

$$||E_A||^2 = \sum_{i=1}^{I} \sum_{j=1}^{J} \sum_{k=1}^{K} e_{ijk}^2$$

where $e_{ijk} = x_{ijk} - \sum_{p=1}^{P} \sum_{q=1}^{Q} \sum_{r=1}^{R} g_{pqr} a_{ip} b_{jq} c_{kr}$ is the residual error for each element (i, j, k) of the tensor. By minimizing $||E_A||^2$ we can be sure that the reconstructed tensor is as close to the original tensor X as possible.

Alternating Least Squares (ALS) Algorithm

ALS is an iterative algorithm used to minimize the objective function $||E_A||^2$ by finding the optimal parameter matrices A, B, C, and G. The steps are:

1. **Initialization**: Begin by initializing the matrices A, B, C, and G with random values or using some other heuristic (using normal, uniform distribution or using SVD).
2. **Update Each Matrix Alternately**:
   - **Update** A: Keeping B, C, and G fixed, update A to minimize $||E_A||^2$. **Update** B**,** C in the same way, keeping fixed the other matrices. Each of these updates can be solved as a **least-squares problem**.

- The **core array** G can be found conditional on A, B, and C by a simple projection of X onto A, B, and C. In matrix notation this reads:

$$G_A = A^\top X_A (C \otimes B)$$

3. **Repeat**: Continue alternatingly updating A, B, C, and G until the reconstruction error $||E_A||^2$ converges: if the change in $||E_A||^2$ between two successive iterations is below a predefined threshold, the algorithm stops.
4. **Check for Local Minima**: Since ALS can converge to a local minimum, it's recommended to run the algorithm multiple times with different starting points for the parameter matrices.

The CHull Method

In general, no a priory information is available regarding the optimal rank *(P,Q,R)* underlying a Tucker3 model for a dataset at hand. The **CHull method** is an heuristic model selection technique that helps to find the optimal rank (P, Q, R) of a Tucker3 decomposition for a dataset by balancing model fit and model complexity. Usually, researchers analyze a range of models from the simplest $(1, 1, 1)$ to a maximum rank, $(P_{\max}, Q_{\max}, R_{\max})$, and the CHull method helps identify the model with an optimal trade-off between fit and complexity.

Steps of the CHull Method

1. **Model Fit f_i:** it can be measured by the *Sum of Squared Residuals (SSR)* or the *explained variance*.
2. **Model Complexity c_i:** it can be measured by the total number of components, total number of fitted parameters or the number of free parameters.
3. **Plot Fit vs Complexity:** compute the fit f_i and complexity c_i values for each Tucker3 model with ranks ranging from $(1,1,1)$ to $(P_{\max}, Q_{\max}, R_{\max})$ and plot each model as a point (c_i, f_i) on the graph.
4. **St-ratio st_i:** compute **st-ratio** to determine the optimal point:

$$\mathrm{st}_i = \frac{(f_{i-1} - f_i)\,/\,(c_i - c_{i-1})}{(f_i - f_{i+1})\,/\,(c_{i+1} - c_i)}$$

Select the model with the highest st-ratio.

Urban mobility patterns can be analyzed through the data obtained from smart cards. However, the datasets are huge and complex to analyze.

- In most cases, a two-dimensional matrix and techniques such as **principal component analysis (PCA)** and **clustering methods** are used. Nonetheless, with these approaches, there is a significant loss of information, as the original structure of the data is broken.
- Using **tensors**, the original structure of the data can be preserved so the information of different dimensions can be analyzed at the same time.

Definition: Rank-1 Tensor

Definition: A tensor is rank-1 if it can be written as the outer product of vectors, one from each mode:

$$x_{ijk} = a_i b_j c_k, \quad \forall i, j, k$$

or if all the (2×2)-minors of the flattening vanish.

Key Characteristics:

- Simplest possible tensor structure.
- Variability in one mode depends on the others.

PARAFAC: Parallel Factor Analysis

Definition: PARAFAC decomposes a tensor into a sum of rank-1 tensors. Element-wise:

$$x_{ijk} = \sum_{s=1}^{S} a_{is} b_{js} c_{ks} + e_{ijk} \qquad \mathbf{X_A} = \mathbf{A}\mathbf{I_A}(\mathbf{C}^\top \otimes \mathbf{B}^\top) + \mathbf{E_A}$$

where:

- a_{is}, b_{js}, c_{ks} are components for the s-th rank-1 term.
- S is the number of rank-1 components.
- e_{ijk} is the residual error.
- $\mathbf{I_A}$ is the matricized version of the three-way identity array.

Key Features:

- Assumes a strict rank-1 structure for the data.
- Requires $S \cdot (I + J + K)$ parameters.
- Unique under certain conditions (e.g., independent components, low rank).
- Highly interpretable due to simplicity.

Strengths:

- Ideal for additive, rank-1 data structures (in other words, data that can be expressed as a sum of simple, independent patterns).
- Applications: Chemometrics, psychometrics, spectroscopy.

Definition: Tucker3 decomposes a tensor into mode-specific factor matrices and a core tensor. Element-wise:

$$x_{ijk} = \sum_{p=1}^{P}\sum_{q=1}^{Q}\sum_{r=1}^{R} g_{pqr} a_{ip} b_{jq} c_{kr} + e_{ijk} \qquad \mathbf{X}_A = \mathbf{A}\mathbf{G}_A(\mathbf{C}^T \otimes \mathbf{B}^T) + \mathbf{E}_A$$

where:

- a_{ip}, b_{jq}, c_{kr}: Components for the p, q, r-th factors in each mode.
- g_{pqr}: Entries of the core tensor $\mathcal{G}$, capturing interactions.
- P, Q, R: Number of components for each mode.

Key Features:

- Flexible ranks (P, Q, R) for each mode.
- Core tensor $\mathcal{G}$ captures interactions between modes.
- Requires more parameters than PARAFAC:

$$P \cdot I + Q \cdot J + R \cdot K + P \cdot Q \cdot R.$$

- It's not unique: if we apply an orthogonal transformation (rotation), we obtain the same decomposition.

Strengths:

- Ideal for complex datasets with multi-mode interactions.
- Applications: Urban mobility, social networks, image analysis.

Core Tensor Simplification:

- In Tucker3, PARAFAC is obtained by constraining the core tensor $\mathcal{G}$ to be **superdiagonal**:

$$g_{pqr} = 0 \quad \text{unless } p = q = r.$$

- Only diagonal entries g_{sss} contribute, simplifying the decomposition to:

$$x_{ijk} = \sum_{s=1}^{S} g_{sss}\, a_{is}\, b_{js}\, c_{ks} + e_{ijk}.$$

Normalization:

- By setting $g_{sss} = 1$, the model reduces to:

$$x_{ijk} = \sum_{s=1}^{S} a_{is}\, b_{js}\, c_{ks} + e_{ijk}.$$

- So the PARAFAC decomposition is the Tucker3 decomposition with $\mathbf{G}_A = \mathbf{I}_A$.

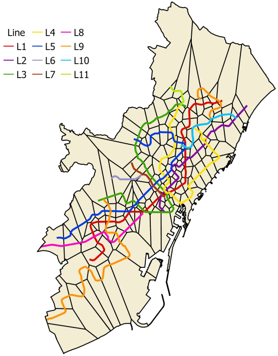
Comparison:

- Tucker3 is flexible but non-unique due to core tensor rotations.
- PARAFAC is stricter and unique under specific conditions.

Data Analysis

Data

The data used correspond to the number of entries in each station from 5 March 2018 to 11 March 2018. This week was chosen because it did not include any public holiday, nor were there extreme weather conditions, so it would reflect the passenger flow under normal conditions. Data are organized in a three-way array $\mathbf{X}$, so that the element x_{ijk} contains the ratio (hourly passengers/total day passengers) of passengers in the station i at time j on day k. In this work, 129 stations (i), 19 time slots (j) and 7 days (k) are analyzed.



Descriptive Analysis

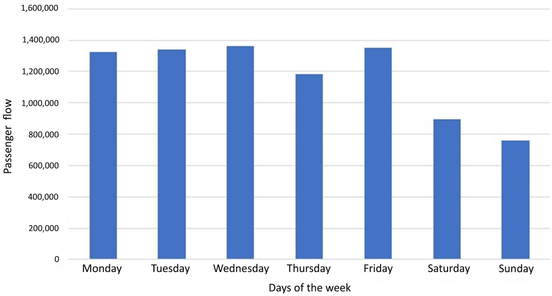


Figure: Daily metro passenger flow

The figure depicts daily metro passenger flow for the study week, showing a noticeable increase on weekdays compared to weekends, suggesting that most passengers are commuters.

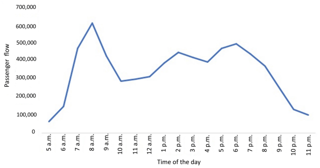


Figure: Hourly passenger flow on workdays

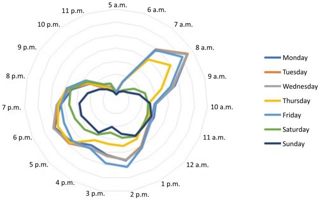


Figure: Total number of passengers per day and hour

Passenger volume varies by time of day, with weekday peak hours occurring at 7–8 AM, 2–3 PM, and 6–7 PM, while weekend peaks are at 1–2 PM and 6–7 PM. Rush hours can differ by station and day. There are significant variations in passenger numbers across stations. To analyze usage patterns and peak hours, data normalization is required, which involves dividing the hourly passenger count by the total number of passengers for each station on that day.

Tucker3 analysis was performed on the dataset for all valid ranks between (1, 1, 1) and (5, 5, 5). In order to select between the many estimated models a solution that optimally balances model fit and model complexity, the **CHull** model selection procedure was applied with the fit percentages and the total number of fitted components (i.e., $P + Q + R$) as complexity value. The model chosen is (2,3,2).

(P, Q, R)	P + Q + R	Fit	St Ratio
$(1,1,1)$	3	86.76	–
$(2,2,2)$	6	94.22	1.06
$(2,3,2)$	7	95.04	8.78
$(3,4,2)$	9	95.82	1.67
$(5,5,4)$	14	96.85	0.92
$(5,5,5)$	15	96.93	–

Figure: Best solutions according to the CHull procedure

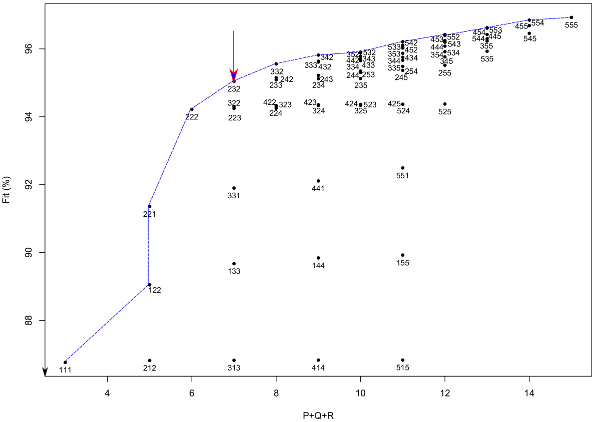


Figure: Plot of the T3 solutions for different values of P, Q, and R

So, the Tucker3 model (by ALS) summarizes the data with matrices A, B, and C, and a core tensor G:

- A: 129×2 matrix (stations): Each row represents a station, and each column corresponds to one of the two **station components**. Each element a_{ij} in this matrix represents how much station i contributes to **component** j.
- B: 19×3 matrix (hours): Each row corresponds to an hour of the day and each column corresponds to one of the three **hour components**. Each element b_{ij} in this matrix represents how much hour i contributes to **component** j.

- C: 7×2 matrix (days): Each row corresponds to a day of the week and each column corresponds to one of the two **day components**. Each element c_{ij} in this matrix represents how much **day** i contributes to **component** j.
- G: $2 \times 3 \times 2$ core tensor, which represents the interaction between the station, hour, and day components. Rows correspond to **station components** (2 rows, one for each component in A). Columns represent combinations of **hour and day components** (6 columns, combining the 3 hour components in B and 2 day components in C).

Stations Matrix

Station	Component 1	Component 2
Roquetes	**0.17**	−0.08
Santa Rosa	**0.17**	−0.08
La Salut	**0.15**	−0.05
Llefià	**0.15**	−0.06
Trinitat Nova	**0.15**	−0.05
Can Cuiàs	**0.14**	−0.03
Can Peixauet	**0.14**	−0.04
Canyelles	**0.14**	−0.04
Ciutat Meridiana	**0.14**	−0.04
Pep Ventura	**0.14**	−0.04
Fira	−0.05	**0.22**
Mas Blau	−0.05	**0.21**
Parc Logístic	−0.05	**0.21**
Palau Reial	−0.03	**0.19**
Ciutadella	−0.03	**0.17**
Hospital de Bellvitge	−0.02	**0.17**
Maria Cristina	−0.01	**0.17**
Mercabarna	−0.02	**0.17**
Diagonal	−0.01	**0.16**
Drassanes	−0.01	**0.16**
Alfons X	0.08	0.05
Av.Carrilet	0.08	0.04
Clot	0.08	0.04
Bellvitge	0.07	0.05
Collblanc	0.08	0.04
Encants	0.08	0.04
Fabra i Puig	0.07	0.05
Hospital de Sant Pau	0.06	0.07
Joanic	0.06	0.06
Poblenou	0.07	0.06

Figure: Component matrix A for selected stations

- **Component 1**: stations in **municipalities**.
- **Component 2**: stations in the **industrial and logistics** zones of the city.

Stations Cluster

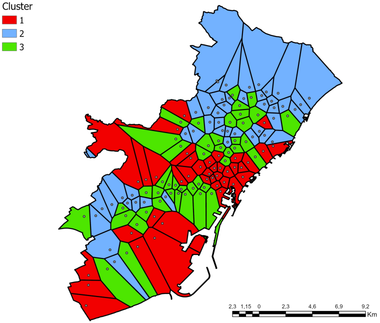


Figure: Map of stations coloured by the cluster they belong to

Cluster 1 (industrial and logistic zones, university and hotels) is formed by stations that have a high score on the second dimension, cluster 2 (municipalities) by those with high scores on the first dimension and cluster 3 (around the city center) by the stations that have high scores in both dimensions.

Hours Matrix

Timetable	Component 1	Component 2	Component 3
5 a.m.	−0.06	0.12	0.05
6 a.m.	0.12	0.18	0.18
7 a.m.	0.06	0.01	**0.62**
8 a.m.	0.05	0.07	**0.68**
9 a.m.	−0.02	**0.25**	**0.29**
10 a.m.	−0.09	**0.49**	0.00
11 a.m.	−0.06	**0.51**	−0.04
12 a.m.	0.03	**0.41**	−0.06
1 p.m.	0.19	**0.27**	−0.03
2 p.m.	**0.37**	0.00	0.06
3 p.m.	**0.31**	−0.02	0.10
4 p.m.	0.14	**0.26**	0.03
5 p.m.	**0.40**	−0.01	0.06
6 p.m.	**0.56**	−0.13	0.00
7 p.m.	**0.33**	0.10	−0.04
8 p.m.	**0.24**	0.14	−0.05
9 p.m.	0.17	0.11	−0.05
10 p.m.	0.09	0.09	−0.05
11 p.m.	0.08	0.07	−0.05

Figure: Component matrix B for timetable

- **Component 1**: Corresponds to **afternoon and evening peak hours**.
- **Component 2**: Reflects **off-peak hours**.
- **Component 3**: Represents the **morning rush hour**.

Some hours, such as 9:00, exhibit high loads across multiple components (e.g., Components 2 and 3).

Days Matrix

Day	Component 1	Component 2
Monday	−0.05	**0.47**
Tuesday	−0.04	**0.47**
Wednesday	−0.02	**0.45**
Thursday	0.03	**0.42**
Friday	0.02	**0.42**
Saturday	**0.63**	0.05
Sunday	**0.77**	0.00

Figure: Component matrix C for days of the week.

- **Component 1**: Corresponds to **weekend days**.
- **Component 2**: Corresponds to **weekdays**.

Core Matrix

	Day Component 1			Day Component 2		
Station Comp.	Hour C1	Hour C2	Hour C3	Hour C1	Hour C2	Hour C3
1	**1.94**	**2.07**	0.77	**2.67**	**2.52**	**3.89**
2	**1.62**	1.44	0.33	**3.36**	**1.87**	1.11

Figure: Core array with $P = 2$, $Q = 3$, and $R = 2$

- The largest variability is captured by $g_{132} = 3.89$ $(P1,\ Q3,\ R2)$, indicating positive interaction between stations with high loadings on the first component and morning peak hours (7-8 a.m.) on weekdays.
- The core element $g_{212} = 3.36$ $(P2,\ Q1,\ R2)$ highlights that stations with high loadings on the second component experience higher passenger proportions during afternoon and evening rush hours on weekdays.

	Day Component 1			Day Component 2		
Station Comp.	Hour C1	Hour C2	Hour C3	Hour C1	Hour C2	Hour C3
1	**1.94**	**2.07**	0.77	**2.67**	**2.52**	**3.89**
2	**1.62**	1.44	0.33	**3.36**	**1.87**	1.11

Figure: Core array with $P = 2$, $Q = 3$, and $R = 2$

- Stations in cluster 2 show lower passenger proportions during peak hours.
- Stations with high loadings across multiple components experience both morning and afternoon peaks.
- Weekend core elements, such as $g_{131} = 0.77$ and $g_{231} = 0.33$, indicate no morning rush hour on weekends, while $g_{121} = 2.07$ suggests the highest interaction occurs during off-peak hours.
- Interaction with the afternoon/evening rush hour is similar across stations ($g_{111} = 1.94$ and $g_{211} = 1.62$).

Thanks for your attention!