The Soul of Wine

Technical Appendix

Mathematical Framework, Algorithms, and Implementation

Jure Skarabot · May 2026

Companion documents: Summary · Technical Appendix · Methods Primer · Data Appendix · Region Reference (Identity, Terroir)

A. Mathematical Framework

A.1 Notation and Setup

Let $\mathcal{R} = \{r_1, r_2, \ldots, r_n\}$ denote the set of $n = 59$ wine regions. For each region r_i the analysis maintains two independent encodings: an *identity encoding* $\mathbf{d}_i \in \mathbb{R}^6$ given by the six expert-scored D-scores $D_1, \ldots, D_6 \in \{-2, -1, 0, +1, +2\}$, and a *terroir encoding* $\mathbf{t}_i \in \mathbb{R}^V$ given by the TF-IDF vectorisation of the Layer 2 terroir narrative over a vocabulary of size $V \approx 800$. The two encodings are derived from disjoint source texts and never share intermediate data structures.

The analytical pipeline produces two clustering partitions of $\mathcal{R}$: an identity partition P_I with $k_I = 6$ clusters and a terroir partition P_T with $k_T = 7$ clusters. The principal analytical question is the agreement between P_I and P_T.

A.2 TF-IDF Vectorisation

For the terroir encoding, each region's six-field Layer 2 narrative is concatenated into a single document. Term frequency–inverse document frequency (TF-IDF) is computed over the corpus of $n = 59$ documents using unigrams and bigrams (n-gram range 1–2), with sublinear TF scaling and a maximum vocabulary size of 800.

For term t in document d of corpus D, the sublinear TF is

$$\mathrm{tf}(t, d) = \begin{cases} 1 + \log f_{t,d} & \text{if } f_{t,d} > 0, \\ 0 & \text{otherwise,} \end{cases}$$

where $f_{t,d}$ is the raw count of t in d. The IDF is

$$\mathrm{idf}(t, D) = \log \frac{|D|}{|\{d \in D : t \in d\}|}.$$

The TF-IDF weight is the product $w_{t,d} = \mathrm{tf}(t, d) \cdot \mathrm{idf}(t, D)$. Each document vector $\mathbf{t}_i$ is L2-normalised after TF-IDF computation. The resulting matrix $T \in \mathbb{R}^{59 \times V}$ is the input to the terroir clustering pipeline.

A.3 Standardisation

Both encodings are standardised before clustering. For each feature j, the standardised value is

$$\tilde{x}_{ij} = \frac{x_{ij} - \mu_j}{\sigma_j}, \qquad \mu_j = \frac{1}{n}\sum_i x_{ij}, \qquad \sigma_j = \sqrt{\frac{1}{n}\sum_i (x_{ij} - \mu_j)^2}.$$

For the identity matrix $D \in \mathbb{R}^{59\times 6}$, standardisation is applied per dimension. For the terroir matrix $T \in \mathbb{R}^{59\times V}$, standardisation is applied per term. Standardisation ensures that all features contribute on a common scale to the subsequent PCA and K-means steps.

A.4 Principal Component Analysis (PCA)

PCA projects each standardised matrix onto an orthonormal basis ordered by explained variance. For a centred data matrix $X \in \mathbb{R}^{n\times p}$, the principal components are the eigenvectors of the sample covariance matrix

$$\Sigma = \frac{1}{n-1} X^\top X.$$

The eigendecomposition $\Sigma \mathbf{v}_k = \lambda_k \mathbf{v}_k$ yields eigenvalues $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_p \geq 0$ and orthonormal eigenvectors $\mathbf{v}_1, \ldots, \mathbf{v}_p$. The projection onto the top m components is $X_{(m)} = XV_{(m)}$ where $V_{(m)} = [\mathbf{v}_1 \cdots \mathbf{v}_m]$. The cumulative explained variance ratio is

$$\rho_m = \frac{\sum_{k=1}^m \lambda_k}{\sum_{k=1}^p \lambda_k}.$$

Identity pipeline. The standardised D-score matrix $\tilde{D} \in \mathbb{R}^{59\times 6}$ is reduced to $m_I = 6$ components, retaining $\rho_6 = 100\%$ of the variance. PCA on the identity matrix is therefore an orthonormal rotation rather than a dimensionality reduction; its purpose is to decorrelate the input dimensions before K-means.

Terroir pipeline. The standardised TF-IDF matrix $\tilde{T} \in \mathbb{R}^{59\times V}$ is reduced to $m_T = 10$ components, retaining $\rho_{10} = 32.2\%$ of the variance. The conservative reduction is necessary because TF-IDF feature spaces are extremely high-dimensional and sparse; clustering directly in the raw 800-dimensional space is dominated by per-term noise.

A.5 K-Means Clustering

K-means partitions n points $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathbb{R}^m$ into k disjoint clusters $C_1, \ldots, C_k$ that minimise the within-cluster sum of squared distances:

$$J(C_1, \ldots, C_k) = \sum_{j=1}^{k} \sum_{\mathbf{x}_i \in C_j} \left\| \mathbf{x}_i - \boldsymbol{\mu}_j \right\|_2^2,$$

where $\boldsymbol{\mu}_j = \frac{1}{|C_j|}\sum_{\mathbf{x}_i \in C_j} \mathbf{x}_i$ is the centroid of cluster C_j. The standard Lloyd algorithm alternates between assignment ($\mathbf{x}_i \to \arg\min_j \left\| \mathbf{x}_i - \boldsymbol{\mu}_j \right\|_2^2$) and centroid update ($\boldsymbol{\mu}_j \to$ mean of assigned points) until convergence.

Algorithm 1 Lloyd's K-means algorithm

Require: Data $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$, cluster count k, n_{init} restarts, max iterations T
Ensure: Cluster assignments $\{c_1, \ldots, c_n\}$ minimising J

for restart $= 1, \ldots, n_{\text{init}}$ **do**
  Initialise centroids $\{\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_k\}$ by k-means++ seeding
  for $t = 1, \ldots, T$ **do**
    Assign: $c_i \leftarrow \arg\min_j \left\|\mathbf{x}_i - \boldsymbol{\mu}_j\right\|_2^2$ for each i
    Update: $\boldsymbol{\mu}_j \leftarrow \frac{1}{|C_j|} \sum_{c_i = j} \mathbf{x}_i$ for each j
    if centroids unchanged **then break**
    end if
  end for
  Record $J^{(\text{restart})}$
end for
return assignments from restart with smallest J

K-means is sensitive to initialisation. The implementation uses k-means++ seeding (Arthur and Vassilvitskii, 2007) and $n_{\text{init}} = 20$ random restarts to mitigate convergence to local minima. The random state is fixed (`random_state` $= 42$) for reproducibility. Pipeline parameters: $k_I = 6$ for identity, $k_T = 7$ for terroir.

A.6 Silhouette Score

For a given partition, the silhouette score (Rousseeuw, 1987) measures cluster cohesion and separation. For each point $\mathbf{x}_i$ assigned to cluster $C(i)$, define

$$a(i) = \frac{1}{|C(i)| - 1} \sum_{\mathbf{x}_j \in C(i),\, j \neq i} \|\mathbf{x}_i - \mathbf{x}_j\|_2 , \qquad b(i) = \min_{C \neq C(i)} \frac{1}{|C|} \sum_{\mathbf{x}_j \in C} \|\mathbf{x}_i - \mathbf{x}_j\|_2 .$$

$a(i)$ is the mean distance from $\mathbf{x}_i$ to other points in its own cluster; $b(i)$ is the mean distance from $\mathbf{x}_i$ to points in the nearest neighbouring cluster. The per-point silhouette is

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}} \in [-1, +1].$$

The overall silhouette is the mean of $s(i)$ over all n points. Values near $+1$ indicate that points are well-matched to their own clusters and poorly matched to neighbouring clusters; values near 0 indicate boundary points; values near -1 indicate likely misassignment.

Empirical silhouette scores: identity partition $S_I = 0.3029$; terroir partition $S_T = 0.2457$. Both exceed the conventional 0.2 threshold for interpretable structure in social-science data.

A.7 The Adjusted Rand Index

The Adjusted Rand Index (Hubert and Arabie, 1985) measures agreement between two partitions of the same set, corrected for chance. Given partitions $P = \{P_1, \ldots, P_a\}$ and $Q = \{Q_1, \ldots, Q_b\}$ of a set of n elements, let $n_{ij} = |P_i \cap Q_j|$ be the count of elements in both P_i and Q_j, and let $a_i = \sum_j n_{ij}$, $b_j = \sum_i n_{ij}$ denote the row and column marginals of the contingency table.

The (unadjusted) Rand Index counts the fraction of pairs of elements that are either co-clustered in both partitions or separated in both partitions. The adjusted form is

$$\mathrm{ARI}(P,Q) = \frac{\displaystyle\sum_{ij}\binom{n_{ij}}{2} - \frac{\sum_i \binom{a_i}{2}\sum_j \binom{b_j}{2}}{\binom{n}{2}}}{\displaystyle\frac{1}{2}\left[\sum_i \binom{a_i}{2} + \sum_j \binom{b_j}{2}\right] - \frac{\sum_i \binom{a_i}{2}\sum_j \binom{b_j}{2}}{\binom{n}{2}}}.$$

Properties: $\mathrm{ARI} = 1$ iff $P = Q$; $\mathbb{E}[\mathrm{ARI}] = 0$ under random partitioning (with marginals fixed); $\mathrm{ARI} < 0$ indicates disagreement worse than chance. ARI is symmetric: $\mathrm{ARI}(P,Q) = \mathrm{ARI}(Q,P)$. ARI does not depend on the cluster labels, only on the partition structure.

For the empirical analysis, $\mathrm{ARI}(P_I, P_T) = 0.023$.

A.8 Chi-Squared Test of Independence

The chi-squared independence test on the $k_I \times k_T$ contingency table $\{n_{ij}\}$ tests the null hypothesis that identity-cluster membership and terroir-cluster membership are statistically independent. Under independence, the expected count in cell (i,j) is

$$e_{ij} = \frac{a_i \cdot b_j}{n}.$$

The test statistic is

$$\chi^2 = \sum_{i=1}^{k_I}\sum_{j=1}^{k_T}\frac{(n_{ij} - e_{ij})^2}{e_{ij}},$$

which under the null follows a chi-squared distribution with $(k_I - 1)(k_T - 1)$ degrees of freedom. For the empirical analysis, $\chi^2 = 38.776$ with $\mathrm{df} = (6-1)(7-1) = 30$, yielding $p = 0.131$. The null of independence is not rejected at $\alpha = 0.05$.

The chi-squared test is sensitive to small expected counts, and the 6×7 contingency table on $n = 59$ regions has many cells with $e_{ij} < 5$. The ARI is therefore the primary independence measure; the chi-squared test is reported as a corroborating diagnostic.

B. Pipeline Parameters

The complete parameter set used by the analysis pipeline is summarised below. All random states are fixed for deterministic, reproducible output.

Parameter	Value
TF-IDF vectoriser	`ngram_range=(1,2)`, `max_features=800`, `sublinear_tf=True`; separate vectoriser per layer
Standardisation	`StandardScaler` applied independently to each input before PCA
PCA (identity)	6 components from 6 D-score dimensions (100% variance retained)
PCA (terroir)	10 components from 800 TF-IDF features (32.2% variance retained)
K-means (identity)	$k = 6$, $n_{\text{init}} = 20$, `random_state` $= 42$
K-means (terroir)	$k = 7$, $n_{\text{init}} = 20$, `random_state` $= 42$; fully model-derived
Evaluation	Silhouette (internal validity), ARI (independence), chi-squared (corroborating)
D-score scale	Integer -2 to $+2$; six dimensions per region, scored from Layer 1 narratives

C. Implementation

C.1 Technology Stack

The Region Affinities tool is implemented as a single-page React application built with Vite. The visualisation layer uses D3.js for force-directed kinship graphs and bipartite flow layouts. The analytical pipeline is implemented in Python (scikit-learn for PCA, K-means, silhouette, ARI, chi-squared; NumPy and SciPy for matrix operations; pandas for data handling). The pipeline output (cluster assignments, PCA coordinates, contingency table, statistics) is serialised to JSON and shipped with the static site.

C.2 Computational Pipeline

Stage 1 (Layer 1 ingestion and D-score scoring). The 59 region identity narratives are read; the primary researcher scores each on the six D-score dimensions. The scored matrix is stored as the canonical D-score table.

Stage 2 (Layer 2 ingestion and TF-IDF vectorisation). The 59 region terroir narratives are read and vectorised independently of Layer 1.

Stage 3 (Standardisation and PCA). Both encodings are independently standardised and projected onto principal components ($m_I = 6$, $m_T = 10$).

Stage 4 (K-means clustering). K-means is run independently on each PCA-projected matrix with the parameters in Section B. Silhouette scores are computed for each partition.

Stage 5 (Independence test). The 6×7 contingency table $\{n_{ij}\}$ is constructed from the two cluster assignments. ARI and chi-squared are computed.

Stage 6 (Visualisation export). Cluster assignments, PCA coordinates, the contingency table, and the test statistics are serialised to JSON. The static site loads this JSON at runtime to render the dual networks, the bipartite flow, the region atlas, and the comparison views.

C.3 Python Reference Implementation

The core analytical pipeline is approximately 60 lines of Python:

```
import numpy as np
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score, adjusted_rand_score
from sklearn.feature_extraction.text import TfidfVectorizer
from scipy.stats import chi2_contingency
from collections import defaultdict

# --- Identity pipeline ---
D = np.array(d_scores)                      # shape (59, 6)
D_std = StandardScaler().fit_transform(D)
D_pca = PCA(n_components=6).fit_transform(D_std)
identity_labels = KMeans(n_clusters=6, n_init=20,
                         random_state=42).fit_predict(D_pca)
S_I = silhouette_score(D_pca, identity_labels)

# --- Terroir pipeline ---
vec = TfidfVectorizer(ngram_range=(1, 2), max_features=800,
                      sublinear_tf=True)
T = vec.fit_transform(layer2_texts).toarray()  # shape (59, ~800)
T_std = StandardScaler().fit_transform(T)
T_pca = PCA(n_components=10).fit_transform(T_std)
terroir_labels = KMeans(n_clusters=7, n_init=20,
                        random_state=42).fit_predict(T_pca)
S_T = silhouette_score(T_pca, terroir_labels)

# --- Independence test ---
contingency = defaultdict(int)
for i in range(len(regions)):
    contingency[(identity_labels[i], terroir_labels[i])] += 1
table = np.zeros((6, 7), dtype=int)
for (i, j), c in contingency.items():
    table[i, j] = c
ari = adjusted_rand_score(identity_labels, terroir_labels)
chi2, p, dof, expected = chi2_contingency(table)
```

C.4 Visualisation Layer

The site renders four primary views. *Dual networks* displays the identity and terroir kinship graphs side by side, with regions as nodes and edges weighted by within-cluster similarity. *Bipartite flow* (Identity ↔ Terroir) renders the contingency table as a Sankey-style flow, making cross-system divergences visually direct. *Region atlas* provides per-region cards showing each region's identity cluster, terroir cluster, D-score profile, and soul metaphor. *Comparison* displays summary statistics: ARI, chi-squared, silhouettes, and per-region divergence indicators. All four views are hydrated from the same JSON output of the Python pipeline; no analytical

computation occurs in the browser.

C.5 Reproducibility

The complete analysis pipeline is available at `analysis/pipeline.py` in the project repository. Running the script reproduces all clustering solutions, PCA coordinates, and statistical tests. Required packages: `scikit-learn`, `numpy`, `scipy`, `pandas`. All random states are fixed (`random_state=42`) for deterministic output. Pipeline outputs (to `data/` directory): `identity_pca.json`, `terroir_pca.json`, `terroir_clusters.json`, `regions.json`, `similarities.json`.

References

- Arthur, D. & Vassilvitskii, S. (2007). k-means++: The advantages of careful seeding. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, 1027–1035.
- Hubert, L. & Arabie, P. (1985). Comparing partitions. *Journal of Classification*, 2(1), 193–218.
- Jolliffe, I. T. (2002). *Principal Component Analysis* (2nd ed.). Springer.
- Lloyd, S. P. (1982). Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 28(2), 129–137.
- Pearson, K. (1900). On the criterion that a given system of deviations from the probable in the case of a correlated system of variables is such that it can be reasonably supposed to have arisen from random sampling. *Philosophical Magazine*, 50(302), 157–175.
- Rousseeuw, P. J. (1987). Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 20, 53–65.
- Salton, G. & Buckley, C. (1988). Term-weighting approaches in automatic text retrieval. *Information Processing & Management*, 24(5), 513–523.