Seasonality Clustering

A Hierarchical Agglomerative Approach

Intern Project – Summer 2025

08/13/2025

Adam Mahmoud

OMNIUM

omniumCPG.com

Background & Context

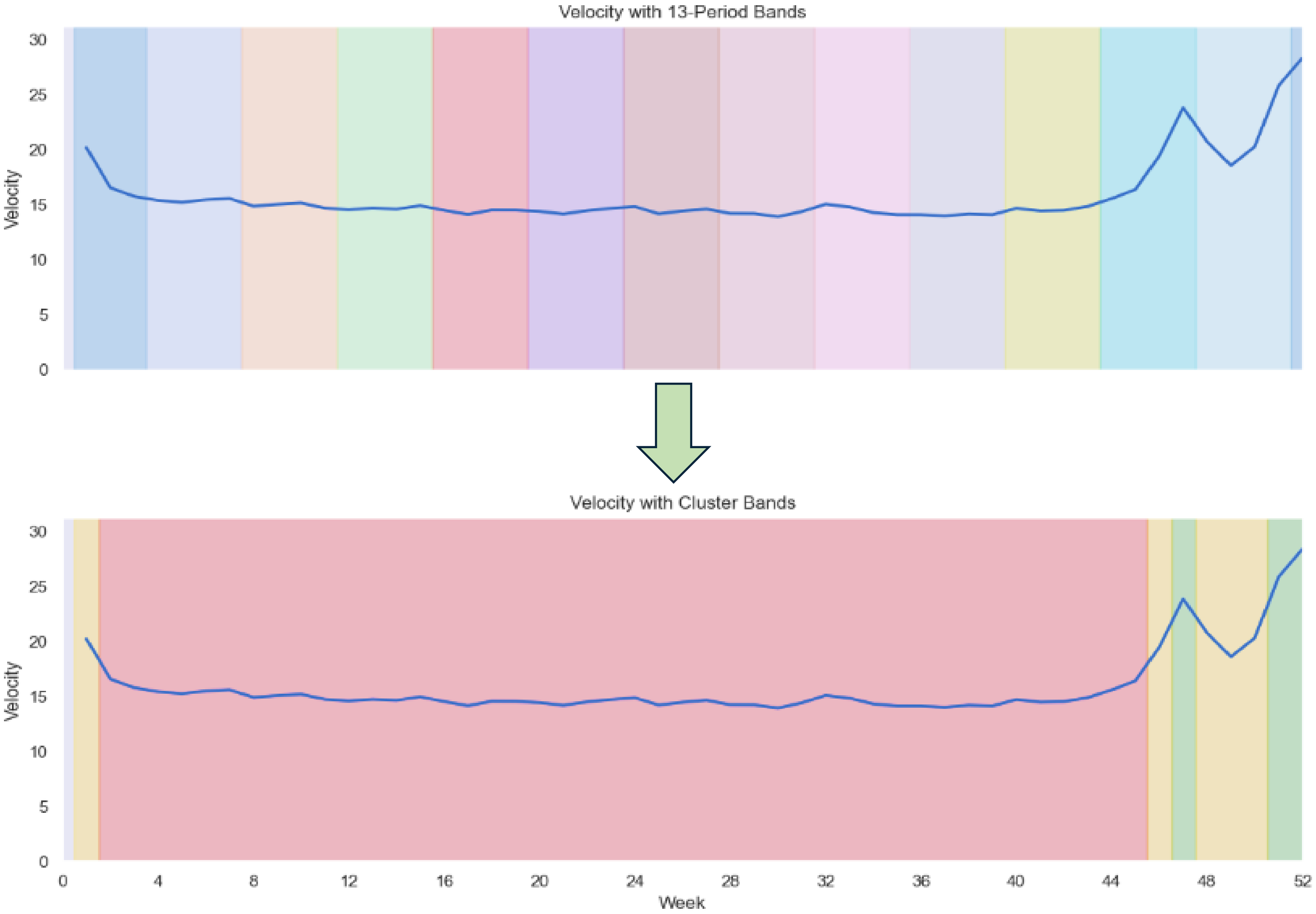
- Current pricing regressions incorporate seasonality through 13 uniform quad periods per year, roughly aligned to holidays.
    - Quad boundaries may misalign with true demand patters → risk of capturing noise instead of seasonality
- Clustering weeks into fewer data-driven seasons can:
    - Better reflect actual demand trends
    - Reduce model complexity & collinearity
    - Improve interpretability
- Bases season definitions on client-specific data

OMNIUM

Executive Summary

- **Problem Statement:** How can we replace fixed quad-period seasonality with data-driven seasonal clusters to reduce pricing regression complexity and capture true demand variation.
- **Hierarchical Agglomerative Clustering**
  - Base Dollar Velocity
  - Base Dollar Velocity & Time
- **Evaluation Metrics**
  - Silhouette Score, Calinski-Harabasz (CH) Index, Davies-Bouldin (DB) Index
  - Dendrogram
  - Silhouette Plot
  - *Velocity & Time Clustering only: Silhouette Score vs. Alpha plots*
- **Regressions**
  - Compare Demand Indices and regression model metrics between pricing regressions that:
    - do not use period
    - use standard quad-periods
    - use clustered periods

Visual Overview: Quad Periods vs. Clustered Seasons – Martinelli's Data



IRI Data, Martinelli's 2022-2024 Weekly (Year-over-Year Aggregated) – Total US - Food

OMNIUM

Data & Attributes

- **Datasets for building clustering pipeline:**
  - Chomps (low seasonality)
  - TruFru (medium seasonality)
  - Martinelli's (high seasonality)
- **Attributes:**
  - Account: Total US Food
  - Base Dollar Velocity: Base Dollars / Stores Selling / Weeks in Distribution
    - Velocity is aggregated over multiple years
    - Note: Database baselines are used

OMNIUM

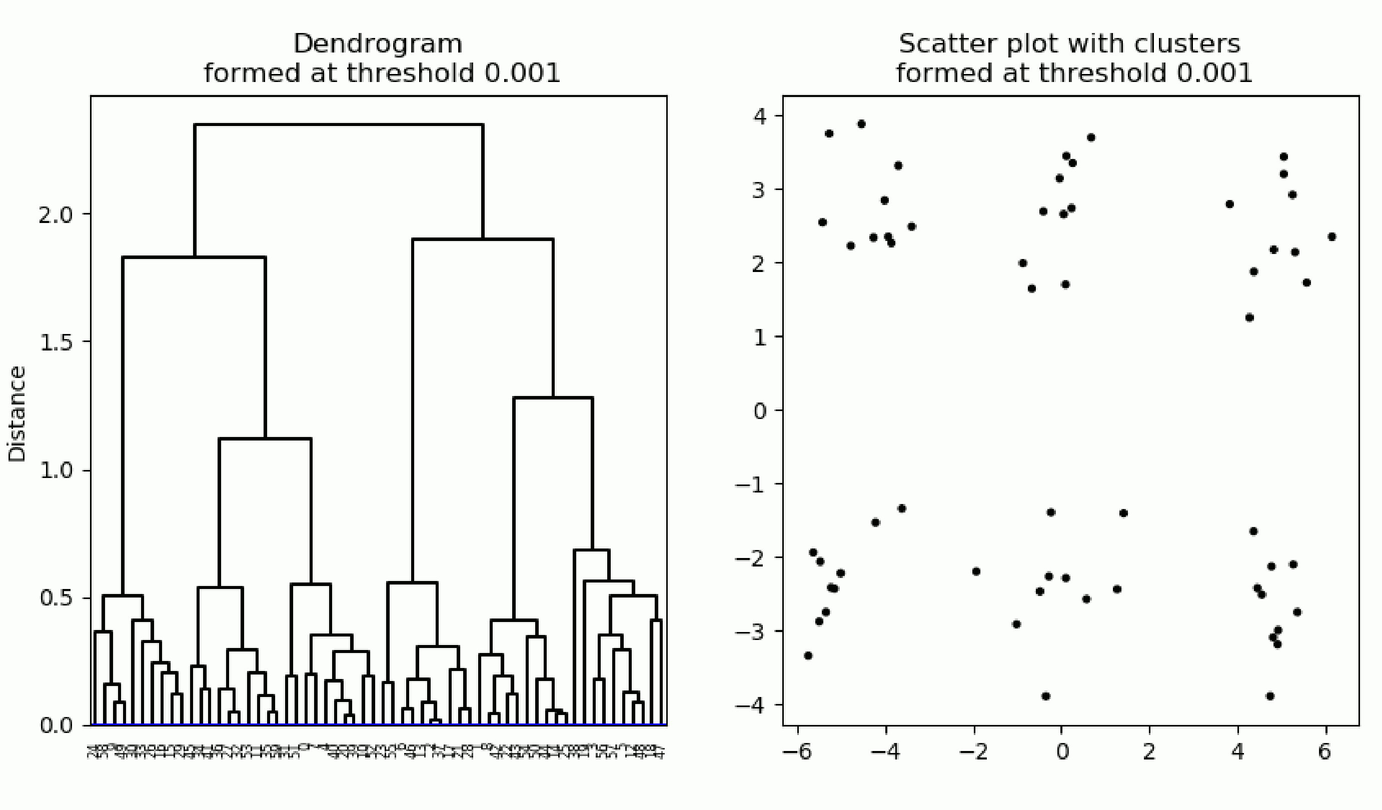
Methodology

- Hierarchical agglomerative clustering
- Clustering on velocity
- Clustering on velocity & time
    - Alpha optimization
- Regression Validation

OMNIUM

Hierarchical Agglomerative Clustering

- **Unsupervised machine learning:** no input-output pairs, no period labels on week numbers
- **Agglomerative:** every data point in its own cluster → merge *similar* pairs of clusters until 1 is left
- **Linkage Criterion:** methods for deciding the order of cluster combinations
  - **Single**, complete, average, weighted, centroid, median, ward



GIF Source: Makkar, *Vast Data Deduction* (students x students, 2022)

Cluster Evaluation Metrics

- Dendrogram: tree diagram showing how clusters merge step-by-step; merge height reflects the distance between joined clusters.

- **Silhouette Score**: measures how similar a point is to its own cluster compared to other clusters. Ranges from -1 (poor fit) to 1 (well separated); higher is better.

- CH Index: ratio of between-cluster variance to within-cluster variance, adjusted for number of clusters. Higher values indicate better defined clusters.

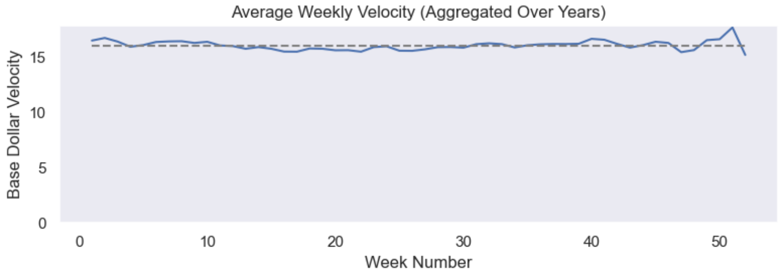
- DB Index: measures average similarity between each cluster and its most similar other cluster; lower values indicate tighter, more distinct clusters.

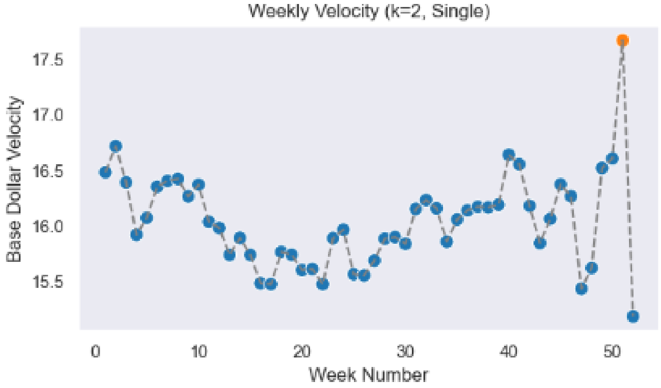
Clustering on Velocity

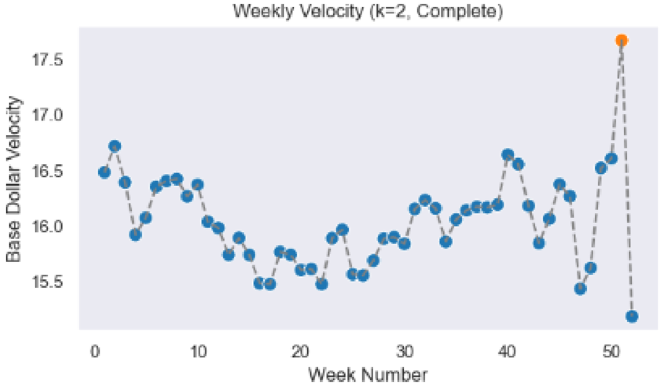
Case Study: Chomps

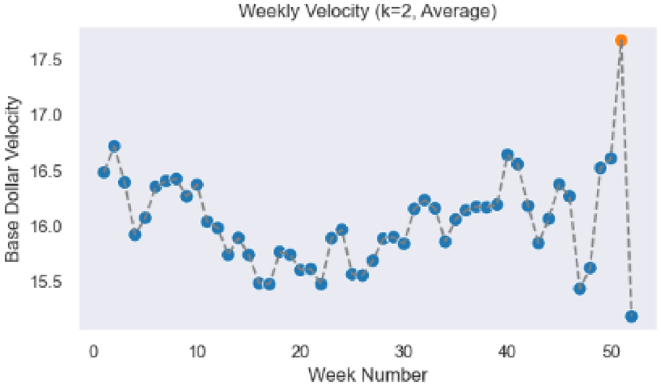
Takeaway: Narrow focus to Single Linkage & Silhouette Score

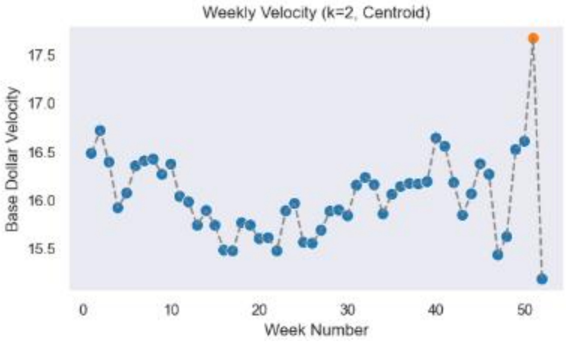
Chomps – Velocity Clustering

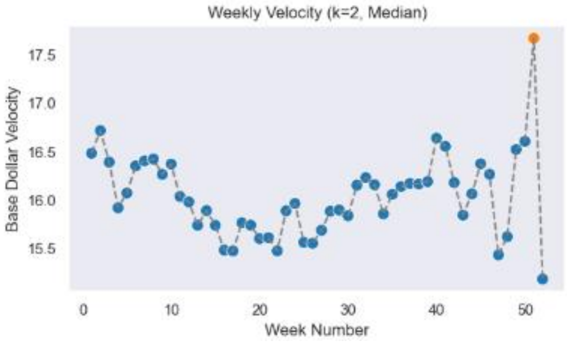


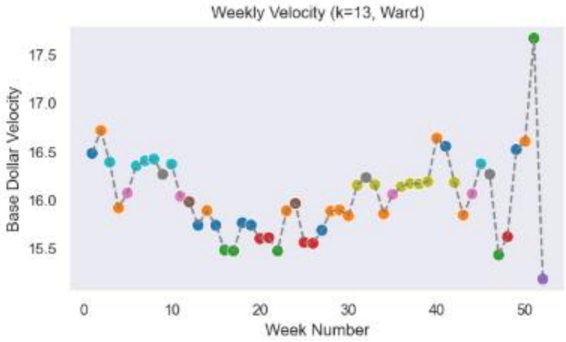






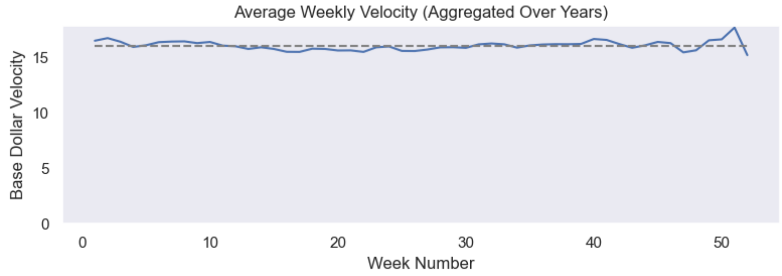


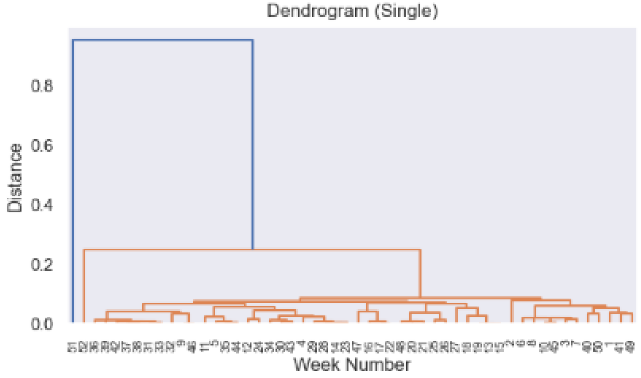


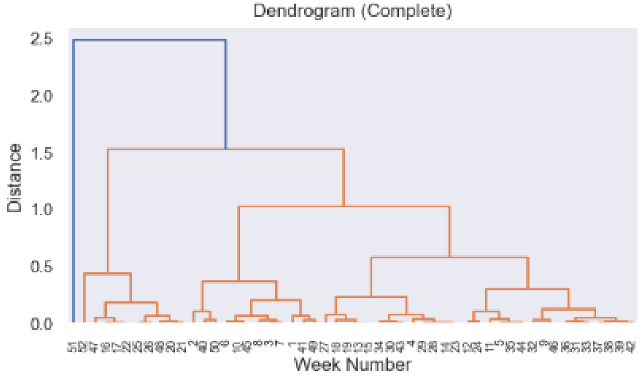


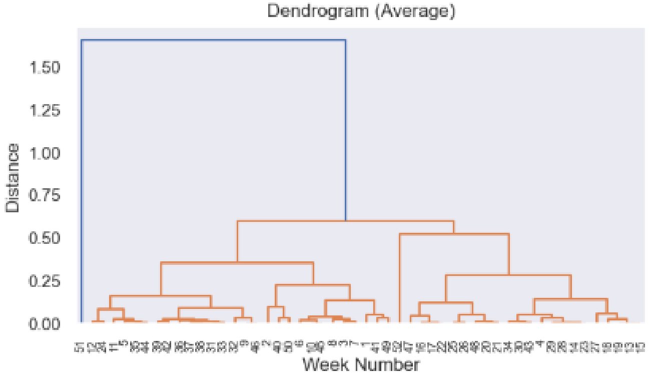
Note: k_max = 13

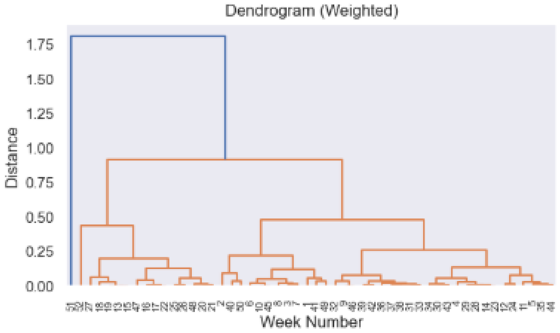
Chomps – Dendrograms

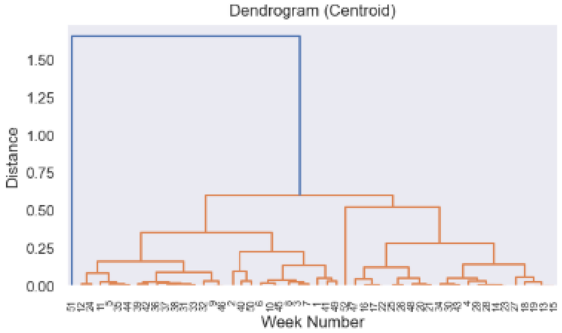


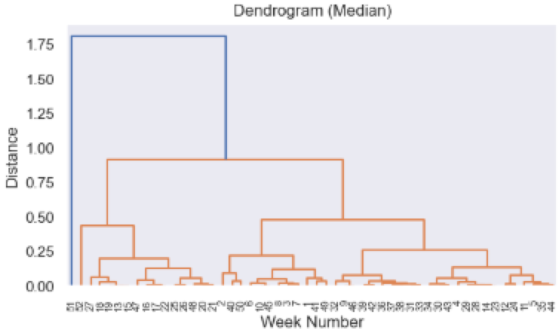


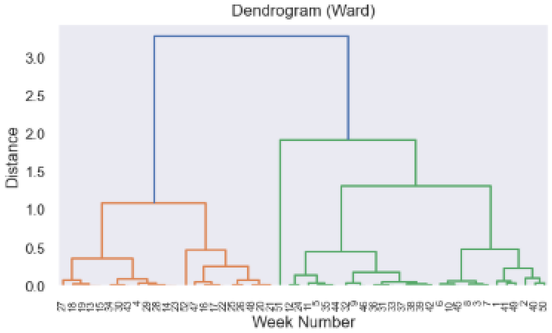






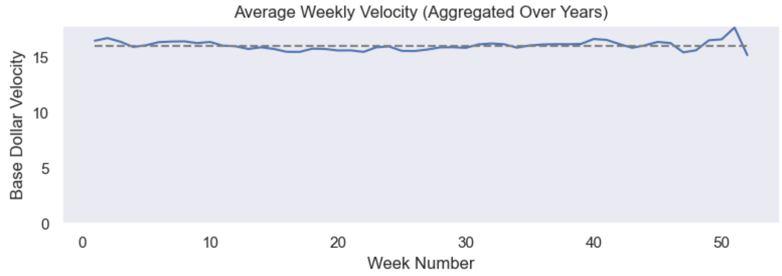


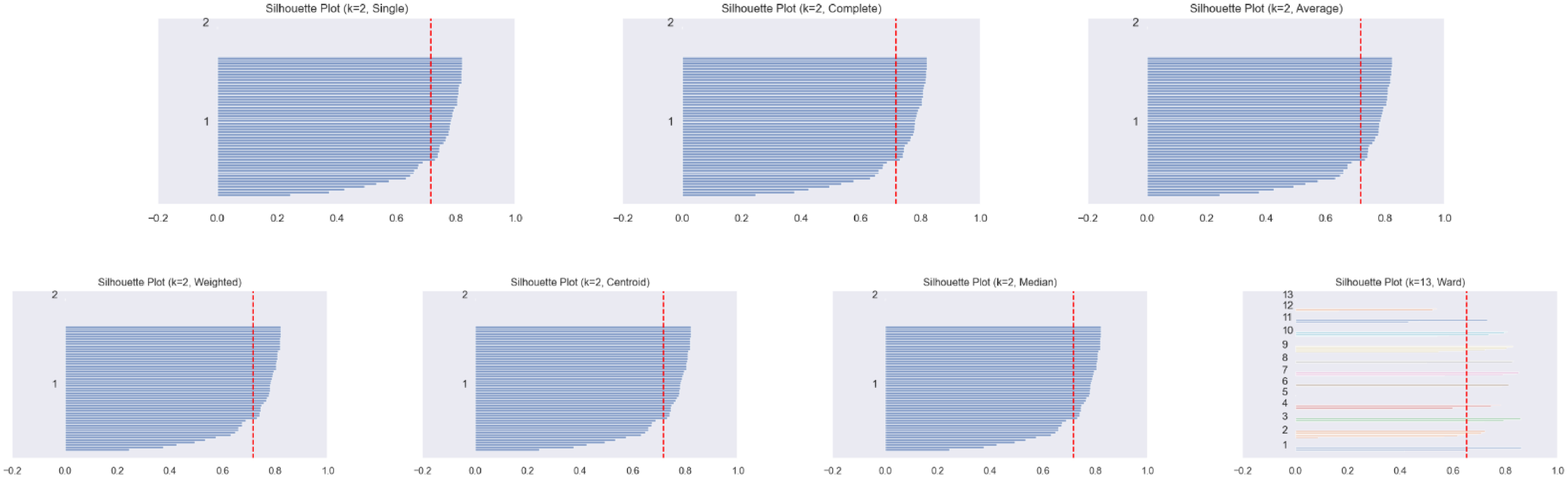




Note: k_max = 13

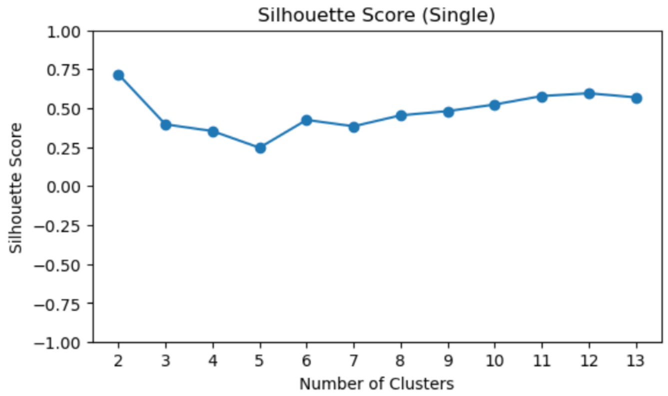
Chomps – Silhouette Plots

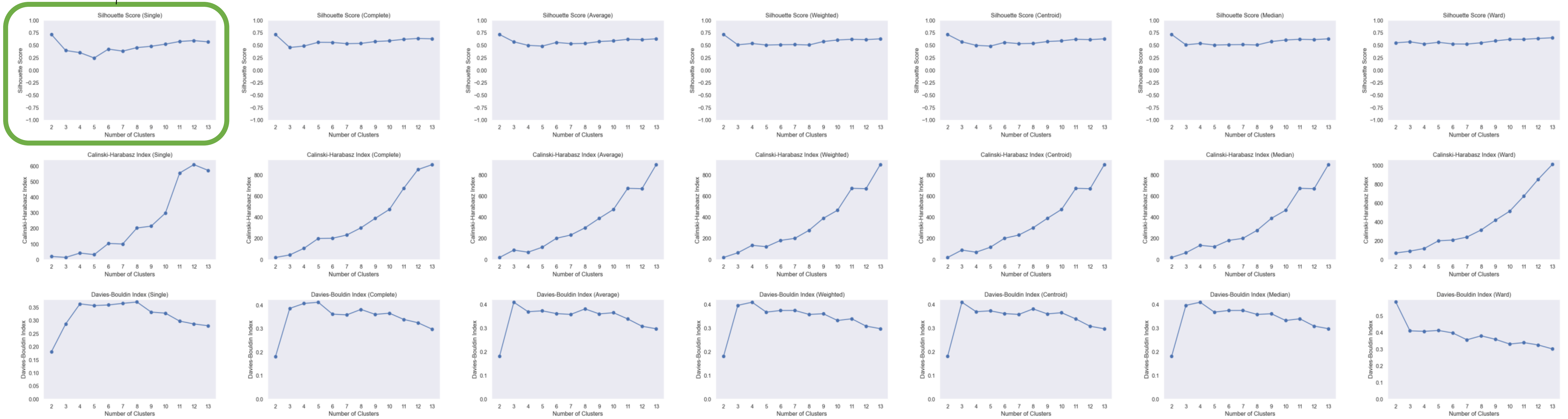




Note: k_max = 13

Chomps – Silhouette Score, CH Index, DB Index by Number of Clusters





Note: k_max = 13

Clustering on Velocity & Time

Case Study: TruFru

Takeaway: Incorporating time in a custom distance function typically worsens clustering

OMNIUM

Custom Distance Functions Tested

❖ **Linear:** Combines normalized week difference and velocity difference with a weighted average

- **Squared Velocity:** Same as linear, but velocity difference is squared to emphasize larger gaps
- **Exponential Decay on Time:** Uses exponential decay for time difference, making nearby weeks much closer
- **Cosine Bump Distance:** Gives extra closeness to weeks within 4 using a cosine curve, then switches to linear growth; piecewise function

OMNIUM

TruFru – Clustering With & Without Time

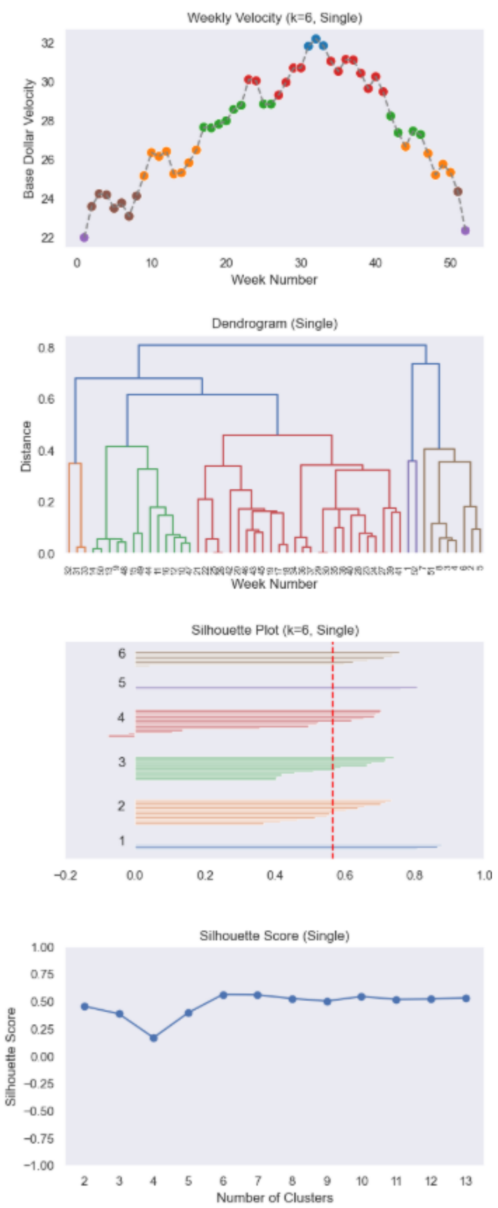
- **Linear Distance Function:**

$$d = \alpha \cdot \frac{\text{week\_diff}}{26} + (1-\alpha) \cdot \frac{|v_1 - v_2|}{v_{\max} - v_{\min}}$$

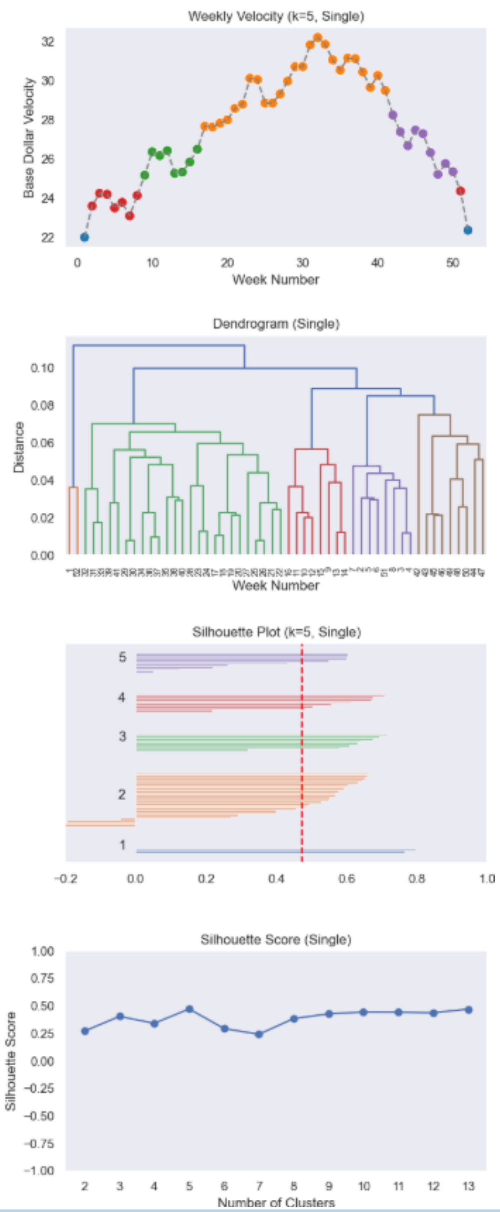
where α = 0.2

- Generally lower silhouette scores
- Clustering with time led to an optimized silhouette score at a lower number of clusters

Without Time



With Time

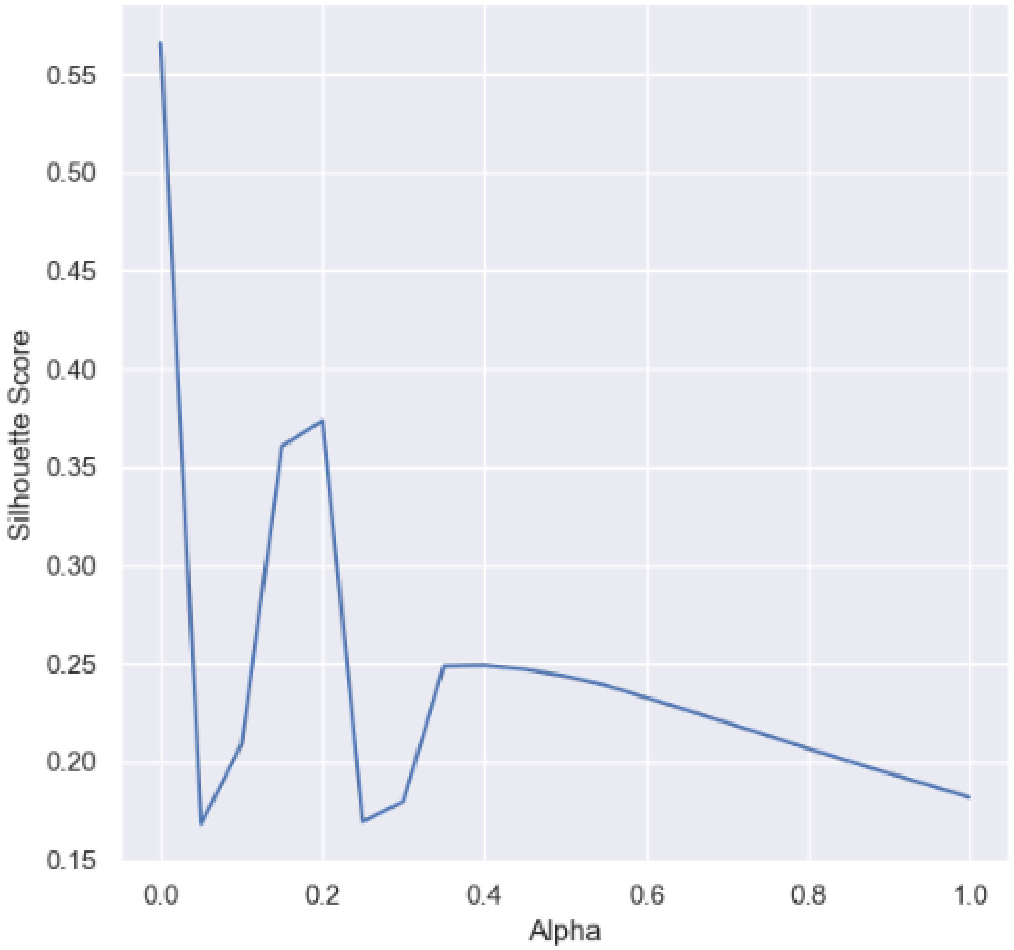


IRI Data, TruFru L156 WE 1/5/25 (Year-over-Year Aggregated) – Total US - Food

OMNIUM

TruFru – Increasing Alpha (putting more weight to the time component in calculating distance) generally worsens average silhouette score

- Linear distance function
- Single linkage
- Number of Clusters = 6



IRI Data, TruFru L156 WE 1/5/25 (Year-over-Year Aggregated) – Total US - Food

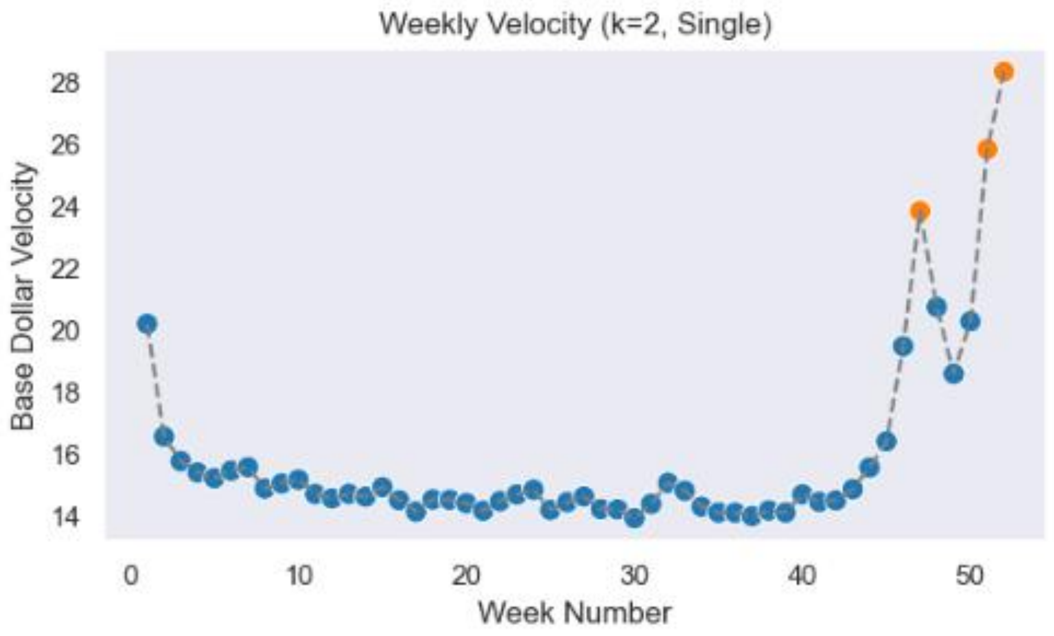
OMNIUM

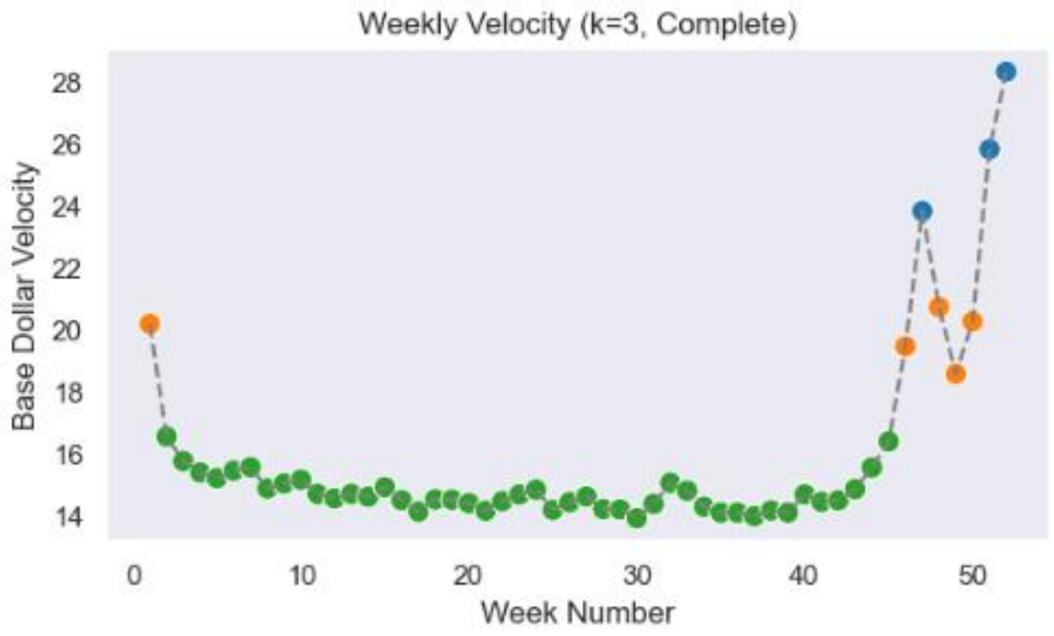
Assessing Results with Pricing Regressions

Case Study: Martinelli's

Takeaway: Clustered seasons can help improve regression R-squared

Martinelli's – Complete linkage appears to cluster seasons more effectively in this case





IRI Data, Martinelli's L156 WE 12/29/24 (Year-over-Year Aggregated) – Total US - Food

OMNIUM

Martinelli's – Clustered periods yield highest R-squared in pricing regression

No Period Factor: np.log(Q('Base Units')) ~ PriceFactor + np.log(Q('ACV')) + AccountFactor

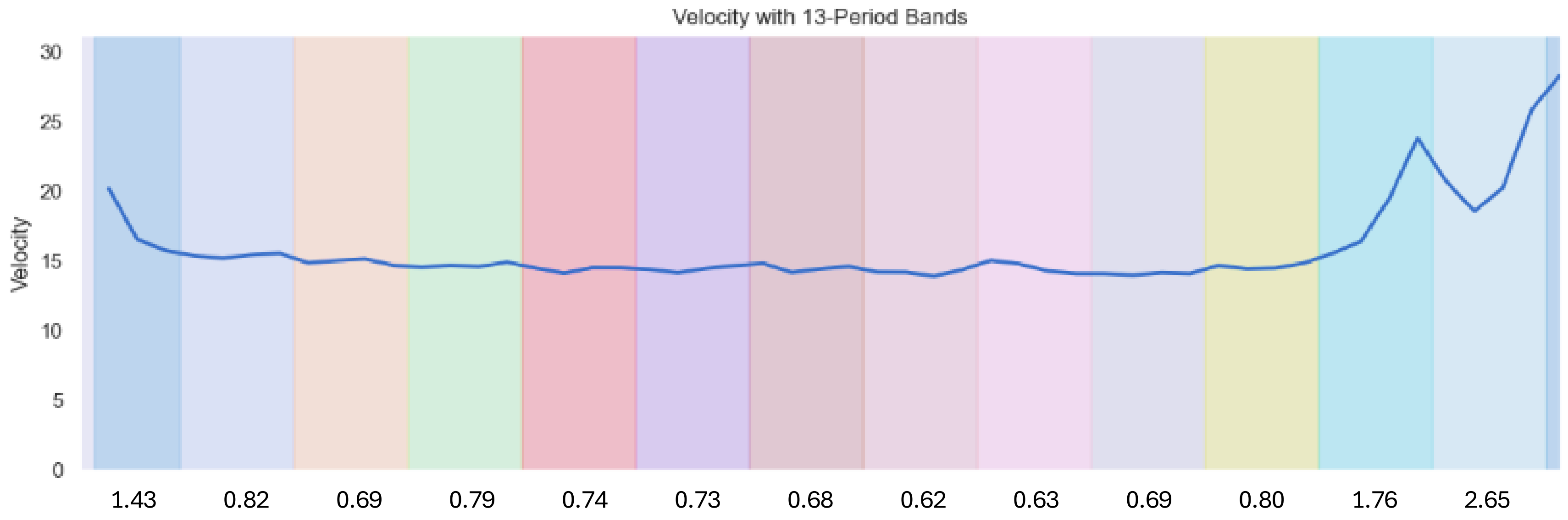
13-Periods Factor: np.log(Q('Base Units')) ~ PriceFactor + np.log(Q('ACV')) + AccountFactor + **PeriodFactor**

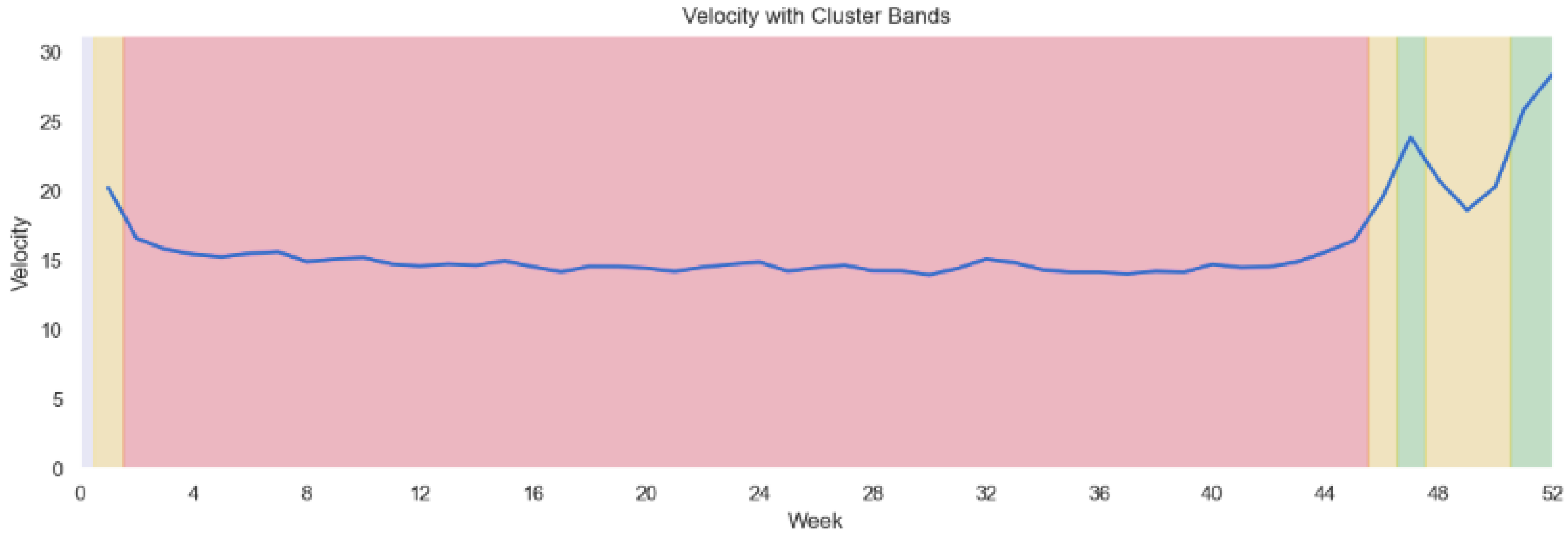
3 Clusters Period Factor: np.log(Q('Base Units')) ~ PriceFactor + np.log(Q('ACV')) + AccountFactor + **ClusteredPeriodFactor**

		Demand Indices		
Price	**Percent of Base Units**	**No Period Factor**	**13-Periods Factor**	**Clustered Periods Factor**
$2.99	16%	1.00	1.00	1.00
$3.49	49%	0.65	0.73	0.84
$3.99	20%	0.37	0.56	0.60
$4.49	7%	0.31	0.51	0.51
$4.99	6%	0.27	0.45	0.46
$5.49	3%	0.25	0.43	0.43
	Constant Elasticity	-2.51	-1.36	-1.62
	R-squared	0.83	0.90	0.94

Regression on single SKU: Martinelli's Gold Medal Apple Sparkling Cider – Glass Bottle, 25.4 oz (1 ct)

IRI Data, Martinelli's L156 WE 12/29/24 – Food Channel Lowers





Seasonality Coefficients

2.07

0.67

0.26

Challenges

- Mapping week numbers across multiple years and aligning holidays consistently
- Many parameters that need to be optimized are at play:
  - Number of clusters
  - Linkage criteria
  - Custom distance metric
  - Alpha value (time vs. velocity weight in chosen custom distance metric)
  - Choice of evaluation metric (Silhouette, CH, DB, regression-based)
- Hard to make a definitive conclusion on impact. Benefits of clustering do not appear to be consistent across datasets and metrics

Continuations

<u>Pipeline</u>

- Implement Python code into Excel for consolidated processing-clustering-regression pipeline.

<u>Regressions</u>

- Determine optimal cluster assignments by maximizing adjusted R-squared or minimizing collinearity metrics (e.g., variance inflation factor, condition number).
- Run the regression pipeline on additional datasets to better quantify the impact of weekly clustering on pricing regressions.
- Compare elasticity predictions from the standard model vs. the clustered model using pre–price-increase data and evaluate which one performs better.

<u>Time-Continuous Seasons</u>

- Explore Markov-constrained clustering to enforce that weeks in the same season are contiguous in time.
- Continue developing and testing custom distance formulas that blend velocity difference with week difference.

<u>Extending Beyond Seasonality – DAFI-Gower Clustering *(Liu et al., 2024)*</u>

- Apply a modified Gower distance that balances the influence of different variable types (numeric, categorical, binary) by scaling them to comparable ranges and weighting them by their importance.
    - Potential variables: velocity, product attributes (package size, flavor, category), promo status, ACV, and account/channel type.

OMNIUM

Acknowledgements

- Sean Dunbar
- William Dumas
- Hazel McCarthy
- Bob Dumas
- Diana Constantinescu

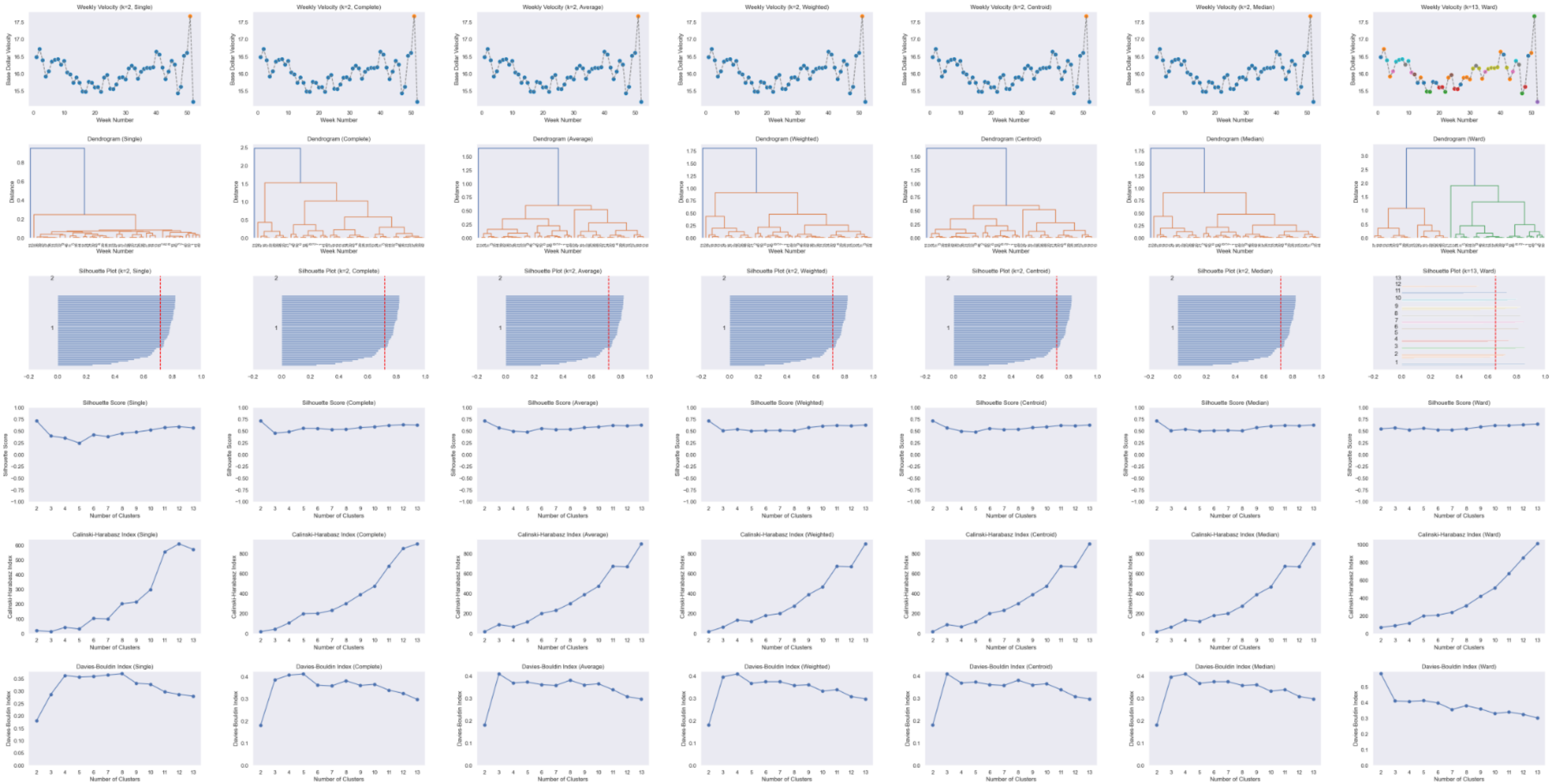
OMNIUM

References

- Liu, P., Yuan, H., Ning, Y. *et al.* A modified and weighted Gower distance-based clustering analysis for mixed type data: a simulation and empirical analyses. *BMC Med Res Methodol* 24, 305 (2024). https://doi.org/10.1186/s12874-024-02427-8
- Makkar, A. (2022, December 10). *Vast Data Deduction: Insight on how a Computer Pinpoints Data*. students x students. Retrieved August 12, 2025, from https://studentsxstudents.com/vast-data-deduction-insight-on-how-a-computer-earmarks-data-af51f9870b58
- Pedregosa, F., *et al.* (2011). *Scikit-learn: Machine Learning in Python*. Journal of Machine Learning Research, 12, 2825–2830. Retrieved August 12, 2025, from https://scikit-learn.org/stable/modules/clustering.html
- Virtanen, P., et al. (2025). *scipy.cluster.hierarchy.linkage* (v1.16.1) [Documentation]. SciPy. Retrieved August 12, 2025, from https://docs.scipy.org/doc/scipy/reference/generated/scipy.cluster.hierarchy.linkage.html

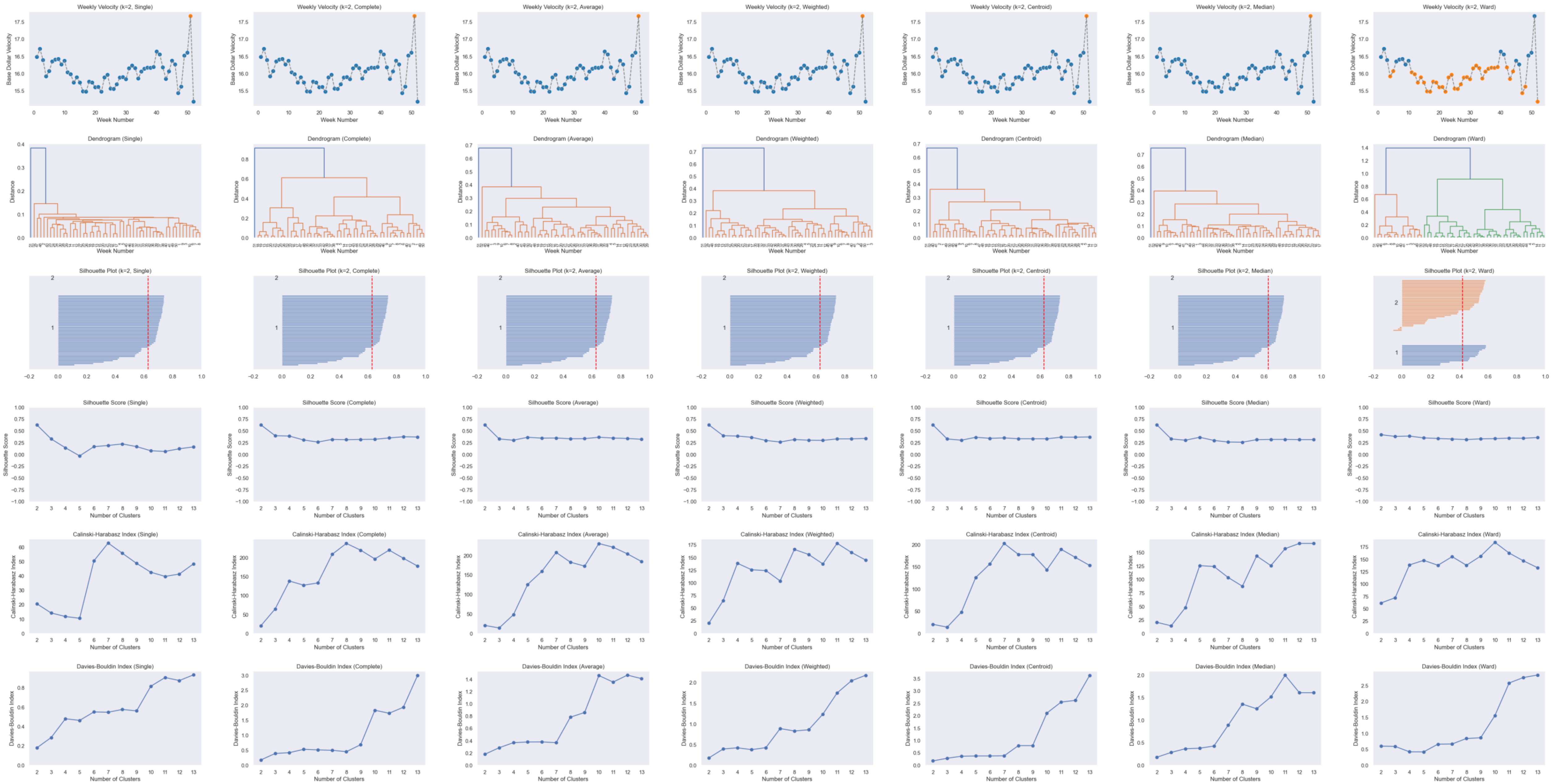
Appendix

OMNIUM


Weekly Velocity (k=2, Single)
Weekly Velocity (k=2, Complete)
Weekly Velocity (k=2, Average)
Weekly Velocity (k=2, Weighted)
Weekly Velocity (k=2, Centroid)
Weekly Velocity (k=2, Median)
Weekly Velocity (k=13, Ward)
Base Dollar Velocity
Week Number
Dendrogram (Single)
Dendrogram (Complete)
Dendrogram (Average)
Dendrogram (Weighted)
Dendrogram (Centroid)
Dendrogram (Median)
Dendrogram (Ward)
Distance
Week Number
Silhouette Plot (k=2, Single)
Silhouette Plot (k=2, Complete)
Silhouette Plot (k=2, Average)
Silhouette Plot (k=2, Weighted)
Silhouette Plot (k=2, Centroid)
Silhouette Plot (k=2, Median)
Silhouette Plot (k=13, Ward)
Silhouette Score (Single)
Silhouette Score (Complete)
Silhouette Score (Average)
Silhouette Score (Weighted)
Silhouette Score (Centroid)
Silhouette Score (Median)
Silhouette Score (Ward)
Silhouette Score
Number of Clusters
Calinski-Harabasz Index (Single)
Calinski-Harabasz Index (Complete)
Calinski-Harabasz Index (Average)
Calinski-Harabasz Index (Weighted)
Calinski-Harabasz Index (Centroid)
Calinski-Harabasz Index (Median)
Calinski-Harabasz Index (Ward)
Calinski-Harabasz Index
Number of Clusters
Davies-Bouldin Index (Single)
Davies-Bouldin Index (Complete)
Davies-Bouldin Index (Average)
Davies-Bouldin Index (Weighted)
Davies-Bouldin Index (Centroid)
Davies-Bouldin Index (Median)
Davies-Bouldin Index (Ward)
Davies-Bouldin Index
Number of Clusters

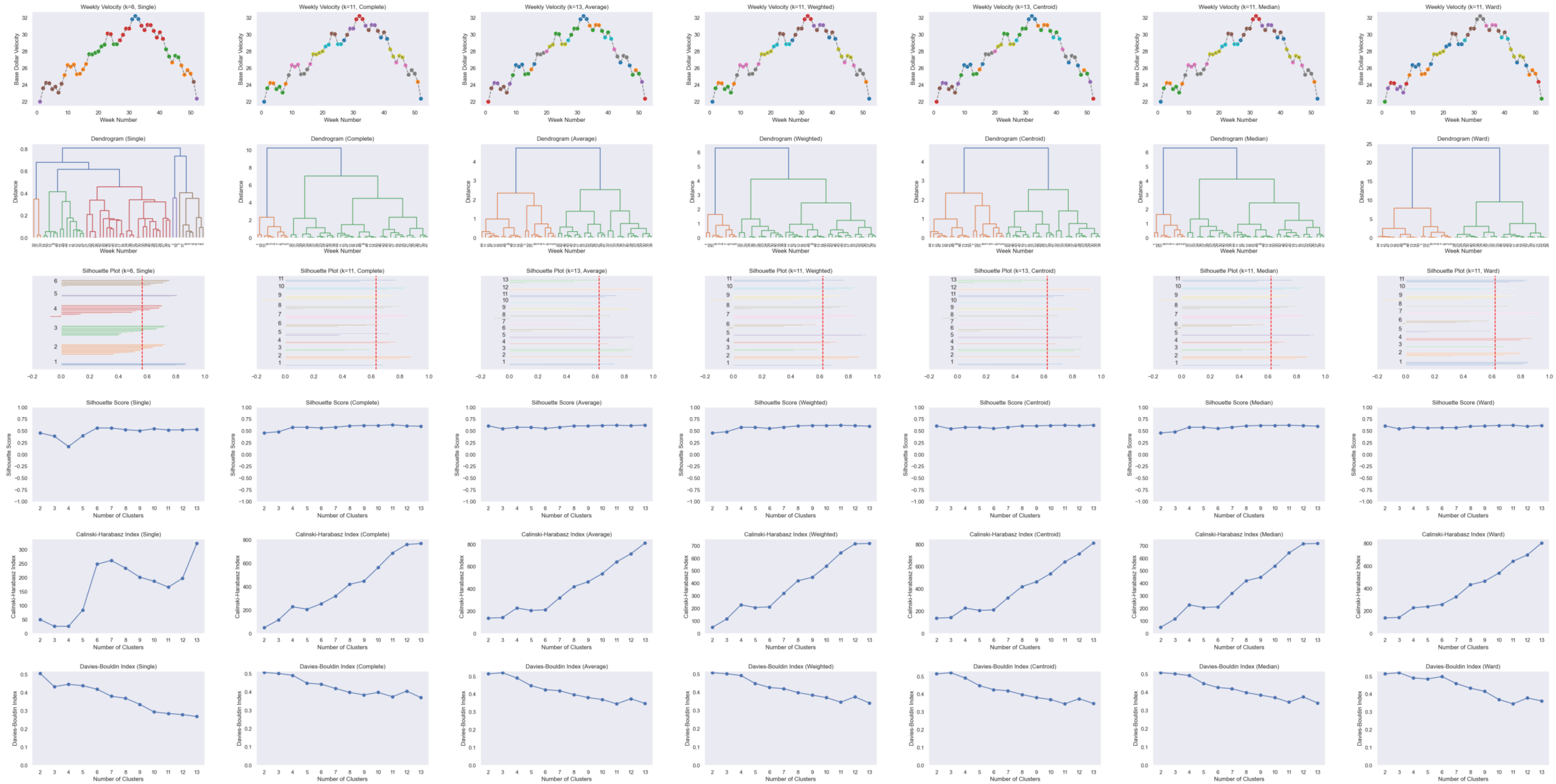
OMNIUM

Chomps – Clustering on Velocity & Time (alpha = 0.1, exp. decay distance function)



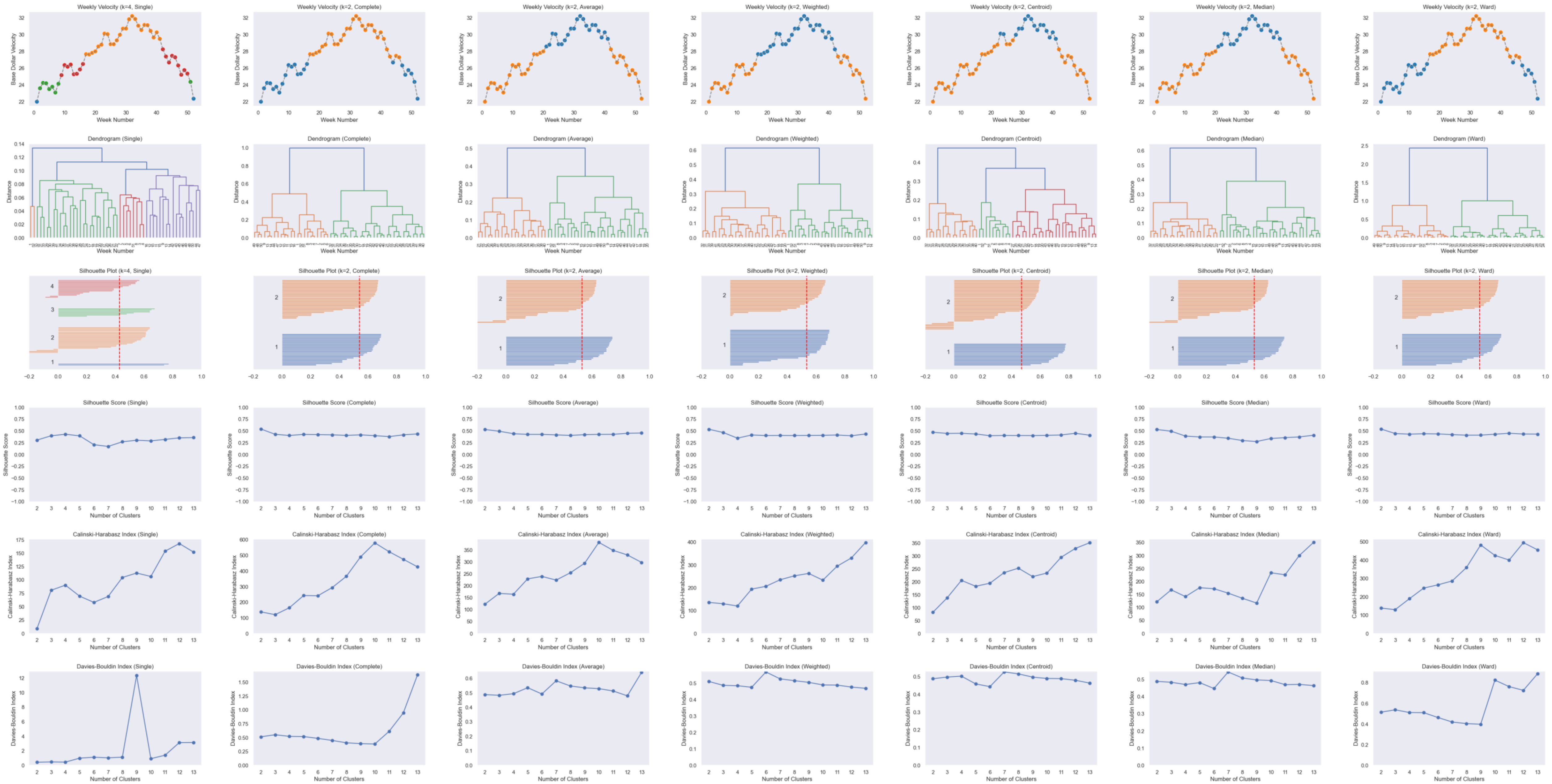
IRI Data, Chomps L156 WE 2/23/25 (Year-over-Year Aggregated) – Total US - Food

OMNIUM

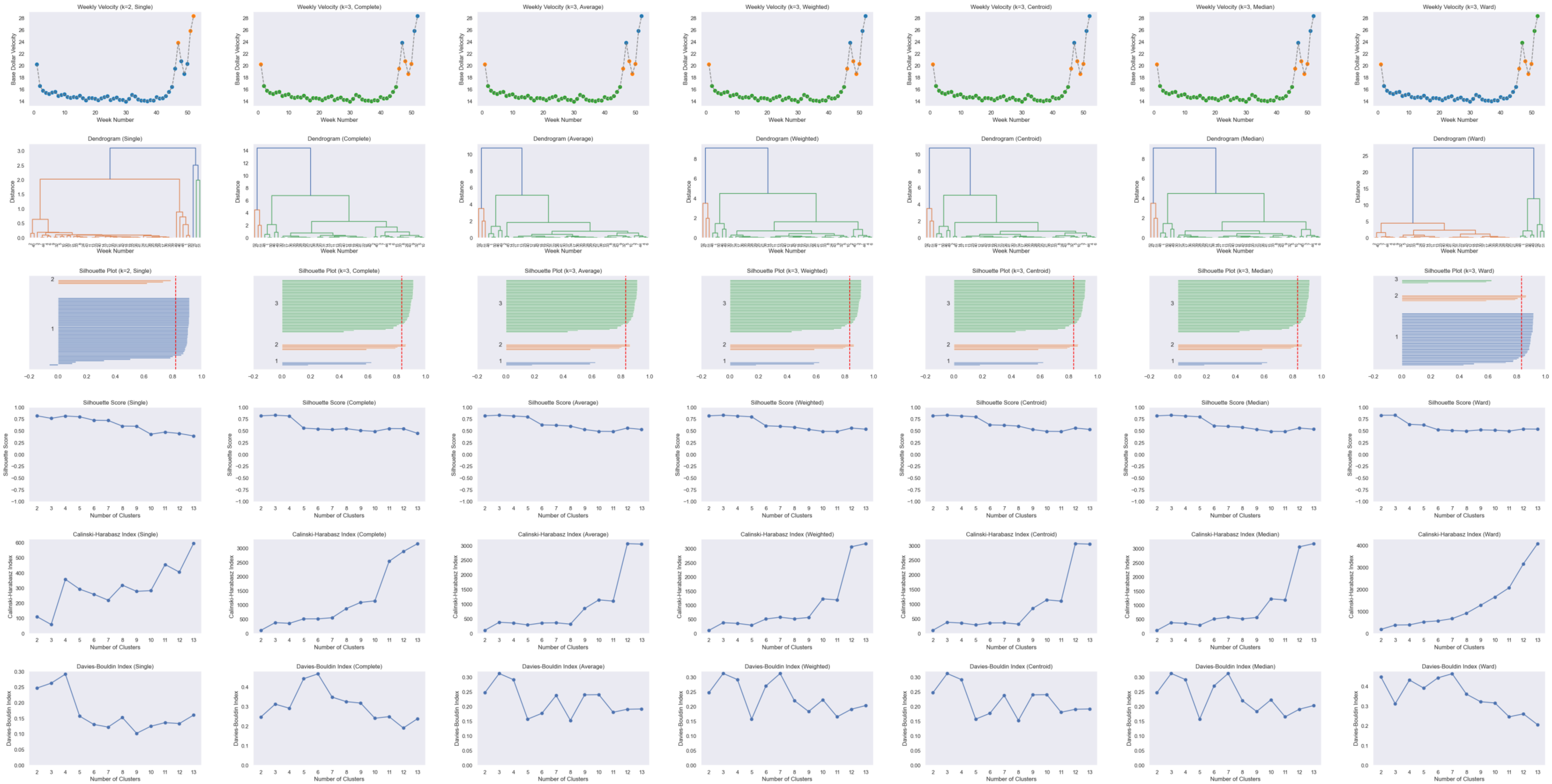


IRI Data, TruFru L156 WE 1/5/25 (Year-over-Year Aggregated) – Total US - Food

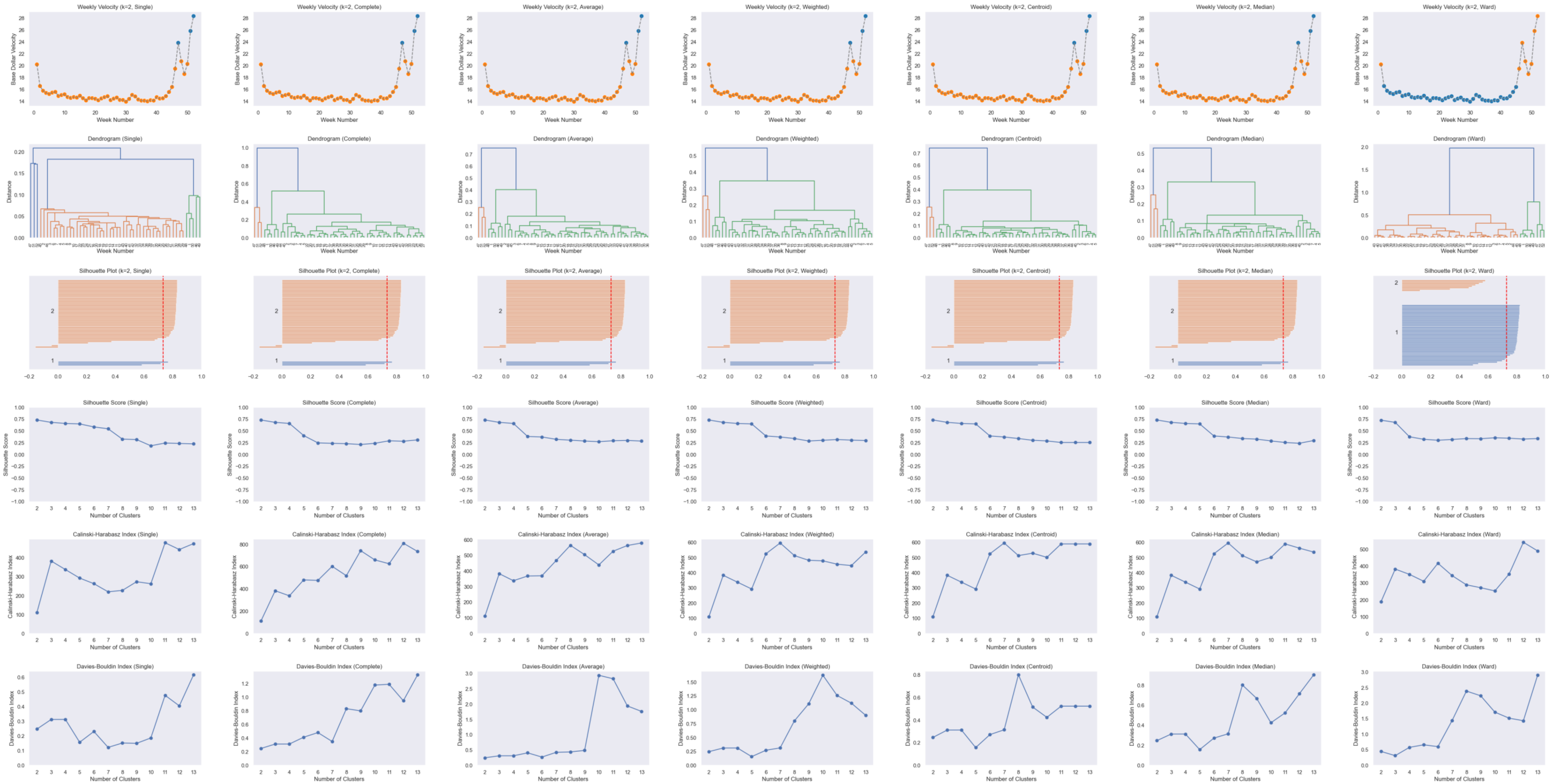
TruFru – Clustering on Velocity & Time (alpha = 0.1, exp. decay distance function)



OMNIUM



IRI Data, Martinelli's L156 WE 12/29/24 (Year-over-Year Aggregated) – Total US - Food



IRI Data, Martinelli's L156 WE 12/29/24 (Year-over-Year Aggregated) – Total US - Food

OMNIUM

Linkage Criterion

- **Single linkage** – Distance between two clusters = shortest distance between any two points (nearest neighbor). Can create “chained” clusters.
- **Complete linkage** – Distance = farthest distance between any two points (furthest neighbor). Produces compact, evenly shaped clusters.
- **Average linkage** – Distance = average of all pairwise distances between points in the two clusters. Balances chaining and compactness.
- **Weighted linkage** – Like average linkage but updates distances with equal weight to each existing cluster, regardless of size.
- **Centroid linkage** – Distance between clusters = distance between their centroids (mean vectors). Can cause reversals in dendrograms.
- **Median linkage** – Similar to centroid but uses median instead of mean for each dimension; more robust to outliers.
- **Ward’s linkage** – Merges clusters that result in the smallest increase in total within-cluster variance; tends to create clusters of similar size.

Linkage Criterion – SciPy Documentation

- method='single' assigns

$$d(u,v) = \min(dist(u[i], v[j]))$$

for all points i in cluster u and j in cluster v. This is also known as the Nearest Point Algorithm.

- method='complete' assigns

$$d(u,v) = \max(dist(u[i], v[j]))$$

for all points i in cluster u and j in cluster v. This is also known by the Farthest Point Algorithm or Voor Hees Algorithm.

- method='average' assigns

$$d(u,v) = \sum_{ij} \frac{d(u[i], v[j])}{(|u| * |v|)}$$

for all points i and j where $|u|$ and $|v|$ are the cardinalities of clusters u and v, respectively. This is also called the UPGMA algorithm.

- method='weighted' assigns

$$d(u,v) = (dist(s,v) + dist(t,v))/2$$

where cluster u was formed with cluster s and t and v is a remaining cluster in the forest (also called WPGMA).

- method='centroid' assigns

$$dist(s,t) = ||c_s - c_t||_2$$

where c_s and c_t are the centroids of clusters s and t, respectively. When two clusters s and t are combined into a new cluster u, the new centroid is computed over all the original objects in clusters s and t. The distance then becomes the Euclidean distance between the centroid of u and the centroid of a remaining cluster v in the forest. This is also known as the UPGMC algorithm.

- method='median' assigns $d(s,t)$ like the `centroid` method. When two clusters s and t are combined into a new cluster u, the average of centroids s and t give the new centroid u. This is also known as the WPGMC algorithm.
- method='ward' uses the Ward variance minimization algorithm. The new entry $d(u,v)$ is computed as follows,

$$d(u,v) = \sqrt{\frac{|v|+|s|}{T} d(v,s)^2 + \frac{|v|+|t|}{T} d(v,t)^2 - \frac{|v|}{T} d(s,t)^2}$$

where u is the newly joined cluster consisting of clusters s and t, v is an unused cluster in the forest, $T = |v| + |s| + |t|$, and $| * |$ is the cardinality of its argument. This is also known as the incremental algorithm.

Source: SciPy v1.16.1 documentation

Cluster Evaluation Metrics – scikit-learn Documentation

Silhouette Score

- **a**: The mean distance between a sample and all other points in the same class.
- **b**: The mean distance between a sample and all other points in the *next nearest cluster*.

The Silhouette Coefficient *s* for a single sample is then given as:

$$s = \frac{b - a}{max(a, b)}$$

The Silhouette Coefficient for a set of samples is given as the mean of the Silhouette Coefficient for each sample.

Source: Source: scikit-learn documentation — *Clustering performance evaluation*

Cluster Evaluation Metrics – scikit-learn Documentation

CH Index

For a set of data E of size n_E which has been clustered into k clusters, the Calinski-Harabasz score s is defined as the ratio of the between-clusters dispersion mean and the within-cluster dispersion:

$$s = \frac{\mathrm{tr}(B_k)}{\mathrm{tr}(W_k)} \times \frac{n_E - k}{k - 1}$$

where $\mathrm{tr}(B_k)$ is trace of the between group dispersion matrix and $\mathrm{tr}(W_k)$ is the trace of the within-cluster dispersion matrix defined by:

$$W_k = \sum_{q=1}^{k} \sum_{x \in C_q} (x - c_q)(x - c_q)^T$$

$$B_k = \sum_{q=1}^{k} n_q (c_q - c_E)(c_q - c_E)^T$$

with C_q the set of points in cluster q, c_q the center of cluster q, c_E the center of E, and n_q the number of points in cluster q.

Source: Source: scikit-learn documentation — *Clustering performance evaluation*

Cluster Evaluation Metrics – scikit-learn Documentation

DB Index

The index is defined as the average similarity between each cluster C_i for $i = 1, \ldots, k$ and its most similar one C_j. In the context of this index, similarity is defined as a measure R_{ij} that trades off:

- s_i, the average distance between each point of cluster i and the centroid of that cluster – also known as cluster diameter.
- d_{ij}, the distance between cluster centroids i and j.

A simple choice to construct R_{ij} so that it is nonnegative and symmetric is:

$$R_{ij} = \frac{s_i + s_j}{d_{ij}}$$

Then the Davies-Bouldin index is defined as:

$$DB = \frac{1}{k} \sum_{i=1}^{k} \max_{i \neq j} R_{ij}$$

Source: Source: scikit-learn documentation — *Clustering performance evaluation*

Custom Distance Functions

```python
def dist_linear(obs_1, obs_2, alpha, min_velocity, max_velocity):
    # linear distances
    week_diff = min(abs(obs_1[0] - obs_2[0]), 52 - abs(obs_1[0] - obs_2[0]))
    week_norm = week_diff / 26
    vel_1 = obs_1[1]
    vel_2 = obs_2[1]
    vel_norm = abs(vel_1 - vel_2) / (max_velocity - min_velocity + 1e-8)
    return alpha * week_norm + (1 - alpha) * vel_norm

def dist_squared(obs_1, obs_2, alpha, min_velocity, max_velocity):
    # squared velocity distance
    week_diff = min(abs(obs_1[0] - obs_2[0]), 52 - abs(obs_1[0] - obs_2[0]))
    week_norm = week_diff / 26
    vel_1 = obs_1[1]
    vel_2 = obs_2[1]
    vel_norm = abs(vel_1 - vel_2) / (max_velocity - min_velocity)
    vel_sq = vel_norm ** 2
    return alpha * week_norm + (1-alpha) * vel_sq
```

```python
def dist_exp_decay(obs_1, obs_2, alpha, min_velocity, max_velocity):
    # exponential decay on time
    week_diff = min(abs(obs_1[0] - obs_2[0]), 52 - abs(obs_1[0] - obs_2[0]))
    tau = 6
    week_decay = 1 - math.exp(-week_diff / tau)
    vel_1 = obs_1[1]
    vel_2 = obs_2[1]
    vel_norm = abs(vel_1 - vel_2) / (max_velocity - min_velocity + 1e-8)
    return alpha * week_decay + (1 - alpha) * vel_norm

def dist_cosine(obs_1, obs_2, alpha, min_velocity, max_velocity):
    # cosine on time within surrounding 4 weeks, then linear
    week_diff = min(abs(obs_1[0] - obs_2[0]), 52 - abs(obs_1[0] - obs_2[0]))
    width = 4
    if week_diff <= width:
        week_bump = 0.5 * (1 - math.cos(math.pi * week_diff / width))
    else:
        week_bump = week_diff / 26
    vel_1 = obs_1[1]
    vel_2 = obs_2[1]
    vel_norm = abs(vel_1 - vel_2) / (max_velocity - min_velocity + 1e-8)
    return alpha * week_bump + (1 - alpha) * vel_norm
```


OMNIUM