Words Apart

Mapping the Hidden Structure of Writing Style Through Multivariate Analysis

Sam Lee Brigg Trendler Daisy Harris

Department of Statistics, Brigham Young University

November 2025

1 Introduction

Writing styles exhibit systematic patterns in pronoun use, temporal framing, reasoning language, and narrative structure, and these patterns can reveal natural groupings of texts beyond their content. Using 1,000 texts scored on eighteen stylistic features (Collins 2003), we investigate whether such features form coherent style-based clusters and how these clusters relate to conventional genre labels. Our analysis proceeds in three steps. We first apply principal component analysis (PCA) to obtain an orthogonal representation of the stylistic variables and generate candidate clusterings for $k = 3, \dots, 7$ using k–means, Gaussian mixture models, and Ward's method. We then evaluate each clustering using MANOVA separation, bootstrap stability, supervised misclassification, and prediction strength to assess distinctness and reproducibility. Finally, we interpret the selected clustering through canonical discriminant analysis and compare the resulting style groups to the original genre categories.

2 Cluster Analysis

We begin by conducting a principal component analysis (PCA) on the 18 linguistic features to obtain an orthogonal basis for clustering. Because the features are measured on heterogeneous scales, PCA is performed on the standardized variables which we define as $\widetilde{\boldsymbol{X}}$. Equivalently, our PCA uses the correlation matrix, rather than the raw covariance matrix[1], ensuring that each feature contributes equally. This is standard practice when variables differ substantially in scale. PCA is obtained by performing a spectral decomposition on the correlation matrix, $\boldsymbol{S}$, where $\boldsymbol{S} = \boldsymbol{CDC'}$. By definition, $\boldsymbol{C} = (\boldsymbol{c}_1, \dots, \boldsymbol{c}_{18})$ is an orthonormal matrix of eigenvectors of $\boldsymbol{S}$ and $\boldsymbol{D} = \operatorname{diag}(d_1, \dots, d_{18})$ is the diagonal matrix of $\boldsymbol{S}$ of eigenvalues ordered $d_1 \geq \cdots \geq d_{18} \geq 0$. The principal component score matrix is then, $\boldsymbol{Z} = \widetilde{\boldsymbol{X}}\boldsymbol{C}$, so that $\boldsymbol{Z}$ provides an orthogonal reparameterization of the standardized feature space. The dth column of $\boldsymbol{Z}$ is the projection of the data onto eigenvector $\boldsymbol{c}_d$ and has variance d_d. Figure 1 displays the cumulative proportion of variance explained by the eigenvalues, showing that the first thirteen components capture approximately 90% of the total variation[2]. Retaining only the principal components that capture the dominant patterns of covariance ensures that clustering operates on the subspace that yields the most separation in $\boldsymbol{X}$, rather than on directions with vanishing variance that inflate Euclidean distances[3]. All subsequent clustering analyses are therefore performed in this reduced PCA space[4].

To estimate a partition of the data for a given k number of clusters that yields optimal cluster separation, we adopt the Calinski–Harabasz (CH) criterion, originally proposed by Caliński and Harabasz (1974), which formalizes cluster quality as a ratio of between-cluster to within-cluster dispersion. For an observation vector $\boldsymbol{x}_i$ with cluster labels $c(i) \in \{1, \dots, k\}$, cluster means $\boldsymbol{\mu}_j$, cluster sizes n_j, and global mean $\boldsymbol{\mu}$, the Calinski–Harabasz index is defined as,

$$\mathrm{CH}(k) = \frac{B_k/(k-1)}{W_k/(n-k)}, \qquad B_k = \sum_{j=1}^{k} n_j \,\|\boldsymbol{\mu}_j - \boldsymbol{\mu}\|^2, \qquad W_k = \sum_{j=1}^{k} \sum_{i \in j} \|\boldsymbol{x}_i - \boldsymbol{\mu}_j\|^2. \tag{1}$$

[1]Notationally, $\boldsymbol{S} = \operatorname{cor}(\boldsymbol{X}) = \frac{1}{n-1}\widetilde{\boldsymbol{X}}'\widetilde{\boldsymbol{X}}$, where where $\widetilde{X}_{ij} = \frac{X_{ij} - \bar{X}_{\cdot j}}{s_j}$, where $\bar{X}_{\cdot j}$ and s_j are the sample mean and standard deviation of the jth variable.

[2]We also look at a scree plot for choosing the number of components to include. We find that five eigenvalues exceed one. Alternative analyses may wish to reduce the covariate space down to five dimensions for this reason. However, for the purposes of our analysis, we keep thirteen dimensions.

[3]Moreover, as shown by Ding and He (2004), the subspace spanned by the leading principal components is the one most aligned with the optimal k-means partition.

[4]That is, we consider the observation vector $\boldsymbol{x}_i \in \mathbb{R}^P$, where $P = 13$. We use the 13 eigenvectors corresponding to the 13 largest eigen values in $\boldsymbol{D}$. We then construct $\boldsymbol{Z} = \widetilde{\boldsymbol{X}}\boldsymbol{C}^{\star}$, where $\boldsymbol{C}^{\star}$ is the corresponding $n \times 13$ matrix of orthonormalized eigenvectors.

Intuitively, CH increases when clusters exhibit high between-cluster separation and low within-cluster dispersion. Because CH is a purely geometric criterion, it provides a model-agnostic benchmark across clustering procedures.

In this study, we propose three unsupervised clustering algorithms. We first implement k–means clustering (Hartigan and Wong 1979). This algorithm partitions the data into k clusters by minimizing the within-cluster sum of squared Euclidean distances. For cluster assignments $c(i)$ and centroids $\boldsymbol{\mu}_j$, the objective function can be defined as,

$$\min_{\{c(i),\boldsymbol{\mu}_j\}} \sum_{j=1}^{k} \sum_{i:c(i)=j} \|\boldsymbol{x}_i - \boldsymbol{\mu}_j\|^2, \tag{2}$$

which is optimized iteratively by assigning each point to the nearest centroid and then repeatedly recomputing centroids as sample means and again until convergence. Because the k–means objective is nonconvex, we employ ten random initializations and record the best solution obtained.

Second, we fit a Gaussian mixture model (GMM) in which, motivated by our use of principal components, each component has a diagonal covariance matrix. This implies that, within each cluster, the coordinates are conditionally independent (under the Gaussian assumption) but may have cluster-specific variances. For the latent cluster label $\zeta_i \in \{1,\ldots,k\}$, we assume,

$$\boldsymbol{x}_i \mid \zeta_i = j \sim \mathcal{N}_P(\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j), \qquad \boldsymbol{\Sigma}_j = \operatorname{diag}(\sigma_{j1}^2,\ldots,\sigma_{jP}^2), \tag{3}$$

so that the component density factorizes as,

$$p(\boldsymbol{x}_i \mid \zeta_i = j) = \prod_{p=1}^{P} \mathcal{N}(x_{ip} \mid \mu_{jp}, \sigma_{jp}^2). \tag{4}$$

Defining π_j as the mixing proportion for component j, with $\sum_{j=1}^{k} \pi_j = 1$. The GMM is fit through the expectation–maximization (EM) algorithm. In the E-step, we compute the posterior cluster probabilities

$$q_{ij} = \Pr(\zeta_i = j \mid \boldsymbol{x}_i) = \pi_j \prod_{p=1}^{P} \mathcal{N}(x_{ip} \mid \mu_{jp}, \sigma_{jp}^2) / \left(\sum_{\ell=1}^{k} \pi_\ell \prod_{p=1}^{P} \mathcal{N}(x_{ip} \mid \mu_{\ell p}, \sigma_{\ell p}^2) \right). \tag{5}$$

In the M-step, we update the parameters using these posterior weights:

$$\pi_j = \frac{1}{n}\sum_{i=1}^{n} q_{ij}, \qquad \mu_{jp} = \frac{\sum_{i=1}^{n} q_{ij} x_{ip}}{\sum_{i=1}^{n} q_{ij}}, \qquad \sigma_{jp}^2 = \frac{\sum_{i=1}^{n} q_{ij} \left(x_{ip} - \mu_{jp}\right)^2}{\sum_{i=1}^{n} q_{ij}}. \tag{6}$$

Final cluster assignments are obtained by the maximum a posteriori rule, $\widehat{\zeta_i} = \arg\max_{1 \le j \le k} q_{ij}$.

Finally, we apply Ward's hierarchical agglomerative method (Ward 1963), which constructs a dendrogram by successively merging the pair of clusters that yields the smallest increase in total within-cluster variance (a reparameterization of Eq. (1)), ensuring that each agglomeration step yields the least distortion to the overall within-cluster dispersion. This method differs fundamentally from the previous two in that it does not optimize over cluster assignments globally but instead builds a full hierarchy of nested partitions.

In addition to assessing separation quality, we also evaluate cluster stability. Namely, we estimate the extent to which the estimated partition is reproducible under perturbations of the dataset. Following a nonparametric bootstrap framework[5], we quantify stability using bootstrap resampling of the Jaccard coefficient. Formally, we define $\{\mathcal{G}_1,\ldots,\mathcal{G}_k\}$ as the k clusters obtained from the original data, where $j \in \{1,\ldots,k\}$ indexes the original clusters. For a given bootstrap resample, we define $\mathcal{S} \subseteq \{1,\ldots,n\}$ as the set of original observations that appear at least once in the resampled dataset (the overlapping set), and let $\{\mathcal{H}_1^*,\ldots,\mathcal{H}_k^*\}$ denote the clusters obtained by reclustering the bootstrap sample, with $s \in \{1,\ldots,k\}$ indexing the bootstrap clusters. For each original cluster $\mathcal{G}_j$, we compute its agreement with each bootstrap cluster $\mathcal{H}_s^*$ on the restricted set $\mathcal{S}$ using the Jaccard index

$$J(\mathcal{G}_j, \mathcal{H}_s^*) = \frac{|(\mathcal{G}_j \cap \mathcal{S}) \cap (\mathcal{H}_s^* \cap \mathcal{S})|}{|(\mathcal{G}_j \cap \mathcal{S}) \cup (\mathcal{H}_s^* \cap \mathcal{S})|}. \tag{7}$$

[5]See Suzuki and Shimodaira (2006) and Hennig (2007) for formal treatments of cluster stability and uncertainty.

Table 1: Clustering quality (Calinski–Harabasz) and bootstrap Jaccard stability by method and number of clusters

	$k = 3$		$k = 4$		$k = 5$		$k = 6$		$k = 7$	
Method	CH	$\bar{\gamma}$ (MCSE)	CH	$\bar{\gamma}$ (MCSE)	CH	$\bar{\gamma}$ (MCSE)	CH	$\bar{\gamma}$ (MCSE)	CH	$\bar{\gamma}$ (MCSE)
k-means	187.69	0.906 (0.0024)	151.85	0.688 (0.0053)	134.31	0.647 (0.0049)	124.27	0.678 (0.0050)	115.27	0.639 (0.0047)
GMM	65.27	0.560 (0.0052)	65.08	0.588 (0.0066)	87.18	0.622 (0.0043)	85.53	0.548 (0.0041)	73.02	0.465 (0.0044)
Ward	157.72	0.608 (0.0031)	128.49	0.509 (0.0026)	112.32	0.477 (0.0040)	103.56	0.487 (0.0042)	94.68	0.459 (0.0033)

Notes: CH denotes the Calinski–Harabasz index computed on the first thirteen principal components. $\bar{\gamma}$ is the mean bootstrap Jaccard stability coefficient, with Monte Carlo standard errors (MCSE) in parentheses, based on 500 nonparametric bootstrap resamples.

The bootstrap stability of $\mathcal{G}_j$ is then defined as the bootstrapped cluster most commensurate with the original clustering on the full data set for that bootstrap iteration: $\gamma_j = \max_{1 \leq s \leq k} J(\mathcal{G}_j, \mathcal{H}_s^*)$. We report the average stability together with Monte Carlo standard errors computed across 500 bootstrap replications in Table 1. Intuitively, larger values of γ indicate clusters that replicate consistently under resampling, whereas smaller values signal unstable clusters that tend to merge, split, or disappear across bootstrap samples. k–means consistently attains the highest CH values[6] and the strongest bootstrap stability. Ward's method yields moderate separation but substantially weaker stability across all k, while the Gaussian mixture model shows lower CH values but competitive stability for $k = 4$ and $k = 5$. However, since k–means attains higher CH scores and more stability across all values of k, we consider clusters formed by this algorithm to be our "best" sets of clusters and will be used for the remainder of the analysis.

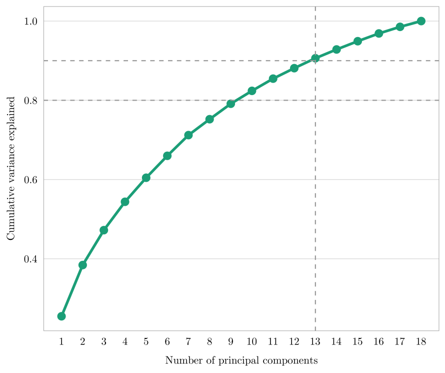

Figure 1: Cumulative variance explained by principal components used for clustering. The vertical dashed line at $k = 13$ denotes the point where $> 90\%$ of the total variation in X is accounted for.

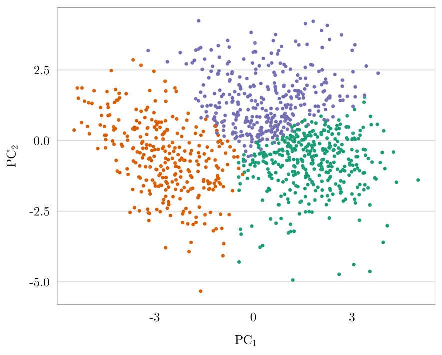

Figure 2: First two principal components plotted against the k-means clustering algorithm with $k = 3$ clusters (shown in different colors).

2.1 Optimal Number of Clusters

To assess how many clusters are supported by the data, we evaluated multivariate separation, supervised classification performance, and resampling-based stability for $k = 3, \ldots, 7$. We present these results in Table 2. We first conducted MANOVA tests to examine whether the multivariate mean vectors of the 13 retained principal components differed across the k–means partitions. For each k, we computed four standard test statistics: Wilks' Λ, Pillai's trace, the Hotelling–Lawley trace, and Roy's greatest root.

All tests were highly significant ($p < 0.0001$), indicating that at least one of the cluster centroids differ across multiple linear combinations of the principal components[7]. The corresponding canonical eigenstructure shows that for smaller k, most separation is concentrated in a single canonical dimension, whereas for larger k, the separation is distributed across several directions. These findings confirm that the PCA features capture meaningful between-group variation and that the resulting k–means partitions represent statistically distinct multivariate clusters. However, statistical significance alone does not determine an optimal k. We

[6]It should be noted that the particular values of the CH scores are generally irrelevant. Because the CH score is a function of total variance, the more principal components we use in our $\boldsymbol{Z}$ matrix, the smaller corresponding values of CH will be.

[7]Again, we stress by construction of the MANOVA tests, as the number of clusters (factors) grows, the probability that we find at least one significant different mean vector (even by chance) artificially increases.

Table 2: Separation metrics for k-means clusters on principal component scores

	$k=3$	$k=4$	$k=5$	$k=6$	$k=7$
Panel A: MANOVA					
Wilks' Λ	0.110	0.056	0.032	0.016	0.011
Pillai's trace	1.288	1.749	2.067	2.548	2.873
Largest canonical root λ_1	3.436	3.380	4.486	4.936	5.074
Percent separation in dim. 1 (%)	76.5	60.9	63.7	59.5	56.8
Canonical corr. ρ_1	0.880	0.878	0.904	0.912	0.914
Panel B: Supervised classification					
Misclass. rate (RF / MN / XGB)	0.23 / 0.48 / 0.10	0.30 / 0.49 / 0.14	0.41 / 0.68 / 0.20	0.41 / 0.86 / 0.21	0.45 / 0.64 / 0.27
Prediction strength $ps(k)$	0.91	0.72	0.64	0.47	0.46

Notes: Entries report separation statistics for k-means clustering on the first thirteen principal components for $k = 3, \ldots, 7$. Panel A reports MANOVA-based measures of between-cluster separation. Panel B reports out-of-sample supervised classification performance for random forest (RF), multinomial logit (MN), and XGBoost (XGB) using 10-fold cross validation; entries in the misclassification row are ordered RF/MN/XGB for each k. All MANOVA tests reject equal mean vectors at $p < 0.0001$ (Wilks' Λ F-tests not shown).

therefore complemented MANOVA with two external validation strategies: supervised misclassification and prediction strength. For each k, we treated the k–means labels as pseudo-outcomes and estimated out-of-sample misclassification rates using 10-fold cross-validation. Each fold held out one observation, trained three classifiers on the remaining data (random forest, multinomial logit, and XGBoost), and predicted the held-out label. This yields a measure of how well cluster structure generalizes. All three classifiers exhibited the same pattern: misclassification was lowest at $k = 3$ and increased monotonically with k. This reflects a well-known drawback of this approach, namely that misclassification necessarily increases as the number of classes grows, producing an implicit penalty on larger k.

To avoid this inflation, we also implemented the prediction strength criterion introduced by Tibshirani and Walther (2005), which evaluates how consistently pairs of points that co-cluster in the training data (X_{tr}) also co-cluster in the test data X_{te}. Formally,

$$\mathrm{ps}(k) = \min_{1 \le j \le k} \frac{1}{n_{k_j}(n_{k_j}-1)} \sum_{i \neq i' \in A_{k_j}} D[C(X_{\mathrm{tr}}, k),\, X_{\mathrm{te}}]_{ii'}\,, \tag{8}$$

where A_{k_j} denotes the set of points assigned to cluster j in the training sample, and $D[\cdot]_{ii'}$ is an indicator of whether i and i' are assigned to the same cluster in both train and test partitions. Higher prediction strength indicates more reproducible clustering. Consistent with the misclassification results, $\mathrm{ps}(k)$ was highest at $k = 3$ and declined steadily thereafter.

Finally, following Ben-Hur et al. (2002), nonparametric bootstrap stability scores reported earlier (Table 1) can provide an additional assessment of robustness. These scores measure how reliably each cluster reappears under bootstrap resampling and re-clustering. Stability was strongest for $k = 3$ and weakened for larger k, aligning with both the supervised and resampling-based evidence. All criteria identify $k = 3$ as the most coherent and reproducible cluster structure. This value of k therefore serves as the basis for our subsequent analysis.

3 Latent Cluster Evaluation

For $k = 3$, the MANOVA produced two nonzero characteristic roots (76.52% and 23.48%), the maximum possible for three groups[8]. The cluster centroids differ along a two–dimensional canonical subspace within the 13-dimensional PCA space, confirming that the three groups are not collinear.

To explore how these clusters differ in practice, we first examined the distribution of genre labels within each group (Figure 3). Cluster 1 is dominated by fiction (86%), forming a stylistically distinct group. Clusters 2 and 3 exhibit more mixed genre compositions: Cluster 2 contains more reporting, skills-and-hobbies, and official communications, while Cluster 3 includes more editorial, biography, and religious writing. To characterize the dimensions underlying this separation, we conducted a linear discriminant analysis (LDA) (Table 3) using the standardized linguistic variables. LDA identifies linear combinations of features that maximize separation among the three clusters. The first discriminant function captures roughly 75% of the discriminant trace and loads most strongly on first–person language, past events, thinking back,

[8]The essential dimensionality is computed as $s = \min(p, k-1) = \min(13, 3-1) = 2$.

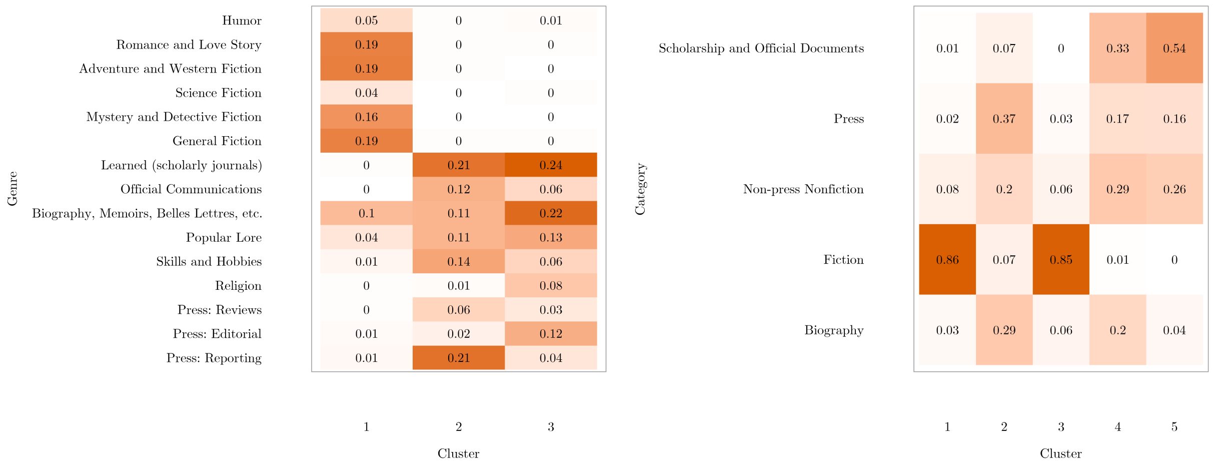

Figure 3: Heatmaps of the genres represented within three clusters (left) and the categories represented within five clusters (right). The intensity of the orange shade represents the proportion of observations within the column cluster belonging to the row (genre).

spatial framing, and linear guidance. These features are most prominent in Cluster 1, consistent with a more personal, narrative, and fiction-oriented style. The second discriminant function emphasizes reasoning, social ties, inner thinking, and negative thinking, with Cluster 3 showing higher scores than Cluster 2. This separation shows the distinction between reflective, introspective writing and more formal, action-oriented prose. We also examined the five-cluster solution and compared it to the broader "super-genre" categories

Table 3: LDA Loadings

LD	FirstPerson	InnerThinking	ThinkPositive	ThinkNegative	ThinkAhead	ThinkBack	Reasoning	Share_SocTies	Direct_Activity
LD1	-0.655	0.060	0.063	0.094	-0.035	-0.360	0.013	0.199	0.039
LD2	0.123	0.492	0.002	0.398	0.082	0.031	0.553	0.622	0.093

LD	Interacting	Notifying	LinearGuidance	WordPicture	SpaceInterval	Motion	PastEvents	TimeInterval	ShiftingEvents
LD1	-0.324	0.137	-0.485	-0.088	-0.411	-0.122	-0.551	0.046	-0.013
LD2	0.165	0.108	0.328	0.003	0.266	-0.148	0.135	0.085	-0.158

Notes: Entries report standardized linear discriminant loadings for LD1 and LD2 based on the 18 linguistic features. Larger absolute values indicate stronger contributions to the discriminant directions; signs indicate direction of association. LD1 accounts for roughly 75% of the total discriminant trace, with LD2 capturing the remaining orthogonal dimension.

(Figure 3, right panel). Two of the clusters are almost entirely fiction, while the remaining clusters mix genres in patterns that do not align with standard category boundaries. This inconsistency suggests that genre labels do not capture the dominant stylistic variation in the data and that clustering based on rhetorical feature may provided a more coherent representation of writing style.

4 Conclusion

In summary, our unsupervised clustering on stylistic features reveals coherent patterns in how texts are written that are only partially captured by conventional genre labels. Writing style, summarized through eighteen linguistic features, exhibits a clear and stable low-dimensional structure. After projecting the data onto principal components and evaluating $k = 3, \ldots, 7$ across Calinski–Harabasz indices, MANOVA separation, supervised misclassification, prediction strength, and bootstrap Jaccard stability, we consistently selected $k = 3$ as the most reproducible partition.

Canonical discriminant analysis further showed that these three clusters differ primarily along two stylistic dimensions. The first corresponds to a personal, narrative style marked by first-person language, past-oriented framing, and spatial or event description, aligning with fiction-heavy texts. The second separates reflective, reasoning-focused writing from more formal, action- or report-oriented prose. Genre labels do not consistently track these stylistic dimensions, particularly in higher-k solutions where clusters mix super-genres. This suggests that stylistic features provide a more coherent representation of writing variation than genre alone.

References

Ben-Hur, A. et al. (2002). "A Stability Based Method for Discovering Structure in Clustered Data". In: *Proceedings of the Pacific Symposium on Biocomputing*, pp. 6–17.

Caliński, T. and J Harabasz (1974). "A dendrite method for cluster analysis". In: *Communications in Statistics* 3.1, pp. 1–27. DOI: `10.1080/03610927408827101`. eprint: `https://www.tandfonline.com/doi/pdf/10.1080/03610927408827101`. URL: `https://www.tandfonline.com/doi/abs/10.1080/03610927408827101`.

Collins, Jeff (2003). "Variations in Written English: Characterizing the Rhetorical Language Choices in the Brown Corpus of Texts". Department of English, Dietrich College of Humanities and Social Sciences. PhD dissertation. Carnegie Mellon University. URL: `https://www.cmu.edu/dietrich/english/academic-programs/phd-rhetoric/phd-rhetoric-dissertation_abstracts/Jeff%20Collins.html`.

Ding, Chris and Xiaofeng He (2004). "Cluster Structure of K-means Clustering via Principal Component Analysis". In: ed. by Honghua Dai et al., pp. 414–418.

Hartigan, J. A. and M. A. Wong (1979). "Algorithm AS 136: A K-Means Clustering Algorithm". In: *Journal of the Royal Statistical Society. Series C (Applied Statistics)* 28.1, pp. 100–108. ISSN: 00359254, 14679876. URL: `http://www.jstor.org/stable/2346830` (visited on 11/17/2025).

Hennig, Christian (2007). "Cluster-wise assessment of cluster stability". In: *Computational Statistics & Data Analysis* 52.1, pp. 258–271. ISSN: 0167-9473. DOI: `https://doi.org/10.1016/j.csda.2006.11.025`. URL: `https://www.sciencedirect.com/science/article/pii/S0167947306004622`.

Suzuki, R. and H. Shimodaira (2006). "Pvclust: An R Package for Assessing the Uncertainty in Hierarchical Clustering". In: *Bioinformatics* 22.12, pp. 1540–1542. DOI: `10.1093/bioinformatics/btl117`.

Tibshirani, Robert and Guenther Walther (2005). "Cluster Validation by Prediction Strength". In: *Journal of Computational and Graphical Statistics* 14.3, pp. 511–528. ISSN: 10618600. URL: `http://www.jstor.org/stable/27594130` (visited on 11/17/2025).

Ward, Joe H. (1963). "Hierarchical Grouping to Optimize an Objective Function". In: *Journal of the American Statistical Association* 58.301, pp. 236–244. ISSN: 01621459, 1537274X. URL: `http://www.jstor.org/stable/2282967` (visited on 11/17/2025).

5 Appendix - R Code

5.1 Data Setup

```r
source("pca.R")

# Read in the data
# Credit: Patric Platts
collins <- read.table(
  "https://tofu.byu.edu/docs/files/stat666/datasets/collins.txt",
  header = TRUE,
  sep = "\t",
  skip = 10, # skip the 10 lines of description before the header
  check.names = FALSE
)

X <- collins[, -c(1, 20:24)]

#PCA on the correlation matrix (this is done by scaling)
X.pca <- pca(X, center = T, scale. = T)
```

5.2 PCA

```r
pca <- function(X, k = NULL, center = TRUE, scale. = FALSE) {
  X <- as.matrix(X)
  n <- nrow(X)
  p <- ncol(X)

  if (is.null(k)) {
    k = min(n - 1, p)
  }

  if (k > min(n - 1L, p)) {
    stop("k must be <= min(n - 1, p).")
  }

  ## center/scale columns
  Xc <- if (center || scale.) {
    scale(X, center = center, scale = scale.)
  } else {
    X
  }

  ## covariance matrix and spectral decomposition
  S <- crossprod(Xc) / (n - 1L) # p x p covariance
  eig <- eigen(S, symmetric = TRUE)

  ## loadings (eigenvectors) and scores (principal components)
  V <- eig$vectors[, 1:k, drop = FALSE] # p x k
  scores <- Xc %*% V # n x k

  colnames(scores) <- paste0("PC", seq_len(k))
  attr(scores, "eigenvalues") <- eig$values[1:k]

  scores # n x k matrix of principal components
}

pca_elbow_plot <- function(
  X,
  k.star = 13,
  max_k = NULL,
  center = TRUE,
  scale. = FALSE,
  target_lines = c(0.80, 0.90)
) {
  X <- as.matrix(X)
  n <- nrow(X)
  p <- ncol(X)

  k_full <- min(n - 1L, p)
  if (is.null(max_k)) {
    max_k <- k_full
  }
  max_k <- min(max_k, k_full)

  pcs <- pca(X, k = k_full, center = center, scale. = scale.)
  eigvals <- attr(pcs, "eigenvalues")

  var_prop <- eigvals / sum(eigvals)
  cum_var_prop <- cumsum(var_prop)

  df <- data.frame(
    k = 1:max_k,
    cum_var = cum_var_prop[1:max_k]
  )

  ggplot(df, aes(k, cum_var, color = "PC")) +
    geom_line(linewidth = 1.2) +
    geom_point(size = 3) +
    geom_hline(
      data = data.frame(y = target_lines),
      aes(yintercept = y),
      linetype = "dashed",
      linewidth = 0.5,
      color = "grey60"
    ) +
    geom_vline(
      aes(xintercept = k.star),
      linetype = "dashed",
      linewidth = 0.5,
      color = "grey60"
    )+
    scale_color_paper(guide = "none") +
    scale_x_continuous(
      breaks = seq_len(max_k),
      minor_breaks = NULL
    ) +
    labs(
      x = "Number of principal components",
      y = "Cumulative variance explained",
      # title = "Elbow Plot for Principal Components",
      # subtitle = "Cumulative proportion of variance explained"
    ) +
    theme_paper(text_size = 64)
}
```

5.3 Calinkski-Harabasz Scores

```r
#Computes the Calinski-Harabasz Score
ch.score <- function(X, G) {
  #X is nxk data matrix
  #G is a vector of group indices corresponding to which groups
  #each row belongs in

  K <- unique(G)

  n <- nrow(X)
  n.j <- tapply(as.data.frame(X), G, nrow)
  mu.j <- tapply(as.data.frame(X), G, colMeans)
  mu <- colMeans(X)

  B.k <- sapply(K, function(k) {
    n.j[k] * crossprod(mu - mu.j[[k]])
  }) |>
    sum()

  W.k <- sapply(K, function(k) {
    m.j <- mu.j[[k]]
    X.g <- X[G == k, ]
    s.g <- apply(X.g, 1, function(x) {
      crossprod(x - m.j)
    }) |>
      sum()
  }) |>
    sum()

  ch.k <- (B.k / (length(K) - 1)) / (W.k / (n - length(K)))
  ch.k
}
```

5.4 K–means Clustering

```r
source("data.R")
source("calinski.R")

library(boot)

kmeans_clusters <- function(X, k, Bstart = 50, seed = NULL) {
  if (!is.null(seed)) {
    set.seed(seed)
  }
  fit <- kmeans(X, centers = k, nstart = Bstart, iter.max = 100)
  fit$cluster
}

kmeans_ch <- function(X, k.grid = 3:7, Bstart = 50, seed = NULL) {
  if (!is.null(seed)) {
    set.seed(seed)
  }

  ch_vec <- sapply(k.grid, function(k) {
    g <- kmeans_clusters(X, k, Bstart = Bstart)
    ch.score(X, g)
  })

  data.frame(k = k.grid, CH = ch_vec)
```

```r
}

kmeans_jaccard_boot <- function(
  X,
  k,
  R = 200,
  Bstart = 50,
  seed = NULL
) {
  if (!is.null(seed)) {
    set.seed(seed)
  }

  n <- nrow(X)
  cl0 <- kmeans_clusters(X, k, Bstart = Bstart)
  boot_fun <- function(data, indices) {
    idx <- indices
    Xb <- data[idx, , drop = FALSE]

    cl_boot <- kmeans_clusters(Xb, k, Bstart = Bstart)

    idx_star <- sort(unique(idx))
    cl0_star <- cl0[idx_star]

    first_pos <- tapply(seq_along(idx), idx, `[`, 1)
    pos_star <- as.integer(first_pos[as.character(idx_star)])
    clb_star <- cl_boot[pos_star]

    gamma_C <- numeric(k)

    for (c in seq_len(k)) {
      C_pts <- idx_star[cl0_star == c]
      if (length(C_pts) == 0L) {
        gamma_C[c] <- 0
      } else {
        jacc_c <- sapply(seq_len(k), function(d) {
          D_pts <- idx_star[clb_star == d]
          if (length(D_pts) == 0L) {
            return(0)
          }
          inter <- length(intersect(C_pts, D_pts))
          union <- length(union(C_pts, D_pts))
          if (union == 0L) 0 else inter / union
        })
        gamma_C[c] <- max(jacc_c)
      }
    }

    gamma_C
  }

  b <- boot(
    data = X,
    statistic = boot_fun,
    R = R,
    parallel = "multicore"
  )

  gamma_bar <- colMeans(b$t)
  gamma_mcse <- apply(b$t, 2, sd) / sqrt(R)

  mean_gamma <- mean(gamma_bar)
  mean_gamma_mcse <- sd(rowMeans(b$t)) / sqrt(R)

  list(
    gamma_bar = gamma_bar,
    gamma_mcse = gamma_mcse,
    mean_gamma = mean_gamma,
    mean_gamma_mcse = mean_gamma_mcse,
    boot_mat = b$t,
    boot_obj = b
  )
}

k.grid <- 3:7

kmeans.summary <- function(X, iters = 500, seed = 666) {
  kmeans_ch_summary <- kmeans_ch(X, k.grid, Bstart = 10, seed = 666)
  print(kmeans_ch_summary)

  kmeans_stab <- lapply(k.grid, function(k) {
    kmeans_jaccard_boot(
      X,
      k = k,
      R = iters,
      Bstart = 10,
      seed = seed
    )
  })
  names(kmeans_stab) <- paste0("k", k.grid)
  print(kmeans_stab)

  kmeans_stab_summary <- data.frame(
    k = k.grid,
    mean_gamma = sapply(kmeans_stab, \(z) z$mean_gamma),
    mean_gamma_mcse = sapply(kmeans_stab, \(z) z$mean_gamma_mcse)
  )

  print(kmeans_stab_summary)
}

kmeans.summary(X = X.pca[, 1:13], iters = 500)

# k        CH
# 1 3 187.6944
# 2 4 151.8520
# 3 5 134.3061
# 4 6 124.2679
# 5 7 115.2718
#    k mean_gamma mean_gamma_mcse
# k3 3  0.9059998     0.002375117
# k4 4  0.6883734     0.005286859
# k5 5  0.6472300     0.004920278
# k6 6  0.6775243     0.005006445
# k7 7  0.6394362     0.004726481

get.data <- function(X, k, seed) {
  as.data.frame(cbind(
    X,
    "label" = kmeans_clusters(X, k, Bstart = 10, seed = seed)
  ))
}

create.data <- function() {
  for (k in k.grid) {
    dat.k <- get.data(X.pca[, 1:13], k, seed = 666)
    readr::write_csv(dat.k, stringr::str_c("Data/dat-", k, ".csv"))
  }
}
```

5.5 Gaussian Mixture Model Clustering

```r
source("data.R")
source("calinski.R")

library(mclust)

gmm_clusters <- function(X, k) {
  fit <- Mclust(X, G = k, modelNames = "VVI")
  fit$classification
}

gmm_ch <- function(X, k.grid = 3:7, seed = NULL) {
  if (!is.null(seed)) {
    set.seed(seed)
  }

  ch_vec <- sapply(k.grid, function(k) {
    g <- gmm_clusters(X, k) #sample(1:k, nrow(X), replace = T)
    ch.score(X, g)
  })

  data.frame(k = k.grid, CH = ch_vec)
}

gmm_jaccard_boot <- function(X, k, R = 200, seed = NULL) {
  if (!is.null(seed)) {
    set.seed(seed)
  }

  n <- nrow(X)
  cl0 <- gmm_clusters(X, k)

  boot_fun <- function(data, indices) {
    idx <- indices
    Xb <- data[idx, , drop = FALSE]

    cl_boot <- gmm_clusters(Xb, k)

    idx_star <- sort(unique(idx))
    cl0_star <- cl0[idx_star]

    first_pos <- tapply(seq_along(idx), idx, `[`, 1)
    pos_star <- as.integer(first_pos[as.character(idx_star)])
    clb_star <- cl_boot[pos_star]

    gamma_C <- numeric(k)

    for (c in seq_len(k)) {
      C_pts <- idx_star[cl0_star == c]
      if (length(C_pts) == 0L) {
        gamma_C[c] <- 0
      } else {
        jacc_c <- sapply(seq_len(k), function(d) {
          D_pts <- idx_star[clb_star == d]
          if (length(D_pts) == 0L) {
            return(0)
          }
          inter <- length(intersect(C_pts, D_pts))
          union <- length(union(C_pts, D_pts))
          if (union == 0L) 0 else inter / union
        })
        gamma_C[c] <- max(jacc_c)
      }
    }

    gamma_C
  }

  b <- boot(
    data = X,
    statistic = boot_fun,
    R = R,
    parallel = "multicore"
  )

  gamma_bar <- colMeans(b$t)
  gamma_mcse <- apply(b$t, 2, sd) / sqrt(R)

  mean_gamma <- mean(gamma_bar)
  mean_gamma_mcse <- sd(rowMeans(b$t)) / sqrt(R)

  list(
    gamma_bar = gamma_bar,
    gamma_mcse = gamma_mcse,
    mean_gamma = mean_gamma,
    mean_gamma_mcse = mean_gamma_mcse,
    boot_mat = b$t,
    boot_obj = b
  )
}

k.grid <- 3:7

gmm.summary <- function(X, iters = 500, seed = 666) {
  gmm_ch_summary <- gmm_ch(X, k.grid, seed = seed)
  print(gmm_ch_summary)
  gmm_stab <- lapply(k.grid, function(k) {
    gmm_jaccard_boot(X, k = k, R = iters, seed = seed)
  })
  names(gmm_stab) <- paste0("k", k.grid)

  gmm_stab_summary <- data.frame(
    k = k.grid,
    mean_gamma = sapply(gmm_stab, \(z) z$mean_gamma),
    mean_gamma_mcse = sapply(gmm_stab, \(z) z$mean_gamma_mcse)
  )
  print(gmm_stab_summary)
}

gmm.summary(X = X.pca[, 1:13], 500)
#   k        CH
# 1 3 65.27188
# 2 4 65.08038
# 3 5 87.18404
# 4 6 85.52907
# 5 7 73.02151
#    k mean_gamma mean_gamma_mcse
# k3 3  0.5604275     0.005184017
# k4 4  0.5876058     0.006601812
# k5 5  0.6217609     0.004314503
# k6 6  0.5483159     0.004132778
# k7 7  0.4651430     0.004417613
```

5.6 Ward Aglomerative Clustering

```
source("data.R")
source("calinski.R")

library(boot)

ward_clusters <- function(X, k) {
  hc <- hclust(dist(X), method = "ward.D2")
  cutree(hc, k = k)
}

ward_ch <- function(X, k.grid = 3:7) {
  hc <- hclust(dist(X), method = "ward.D2")
  ch_vec <- sapply(k.grid, function(k) {
    groups <- cutree(hc, k = k)
    ch.score(X, groups)
  })
  data.frame(k = k.grid, CH = ch_vec)
}

ward_jaccard_boot <- function(X, k, R = 200, seed = NULL) {
  if (!is.null(seed)) {
    set.seed(seed)
  }

  n <- nrow(X)
  cl0 <- ward_clusters(X, k)

  boot_fun <- function(data, indices) {
    idx <- indices
    Xb <- data[idx, , drop = FALSE]

    cl_boot <- ward_clusters(Xb, k)

    #original points that appear in bootstrap sample
    idx_star <- sort(unique(idx))

    #labels of those points in original clustering
    cl0_star <- cl0[idx_star]

    #align bootstrap labels to same original indices
    first_pos <- tapply(seq_along(idx), idx, `[`, 1)
    pos_star <- as.integer(first_pos[as.character(idx_star)])
    clb_star <- cl_boot[pos_star]

    #Jaccard per original cluster
    gamma_C <- numeric(k)

    for (c in seq_len(k)) {
      C_pts <- idx_star[cl0_star == c]
      if (length(C_pts) == 0L) {
        gamma_C[c] <- 0
      } else {
        jacc_c <- sapply(seq_len(k), function(d) {
          D_pts <- idx_star[clb_star == d]
          if (length(D_pts) == 0L) {
            return(0)
          }
          inter <- length(intersect(C_pts, D_pts))
          union <- length(union(C_pts, D_pts))
          if (union == 0L) 0 else inter / union
        })
        gamma_C[c] <- max(jacc_c)
      }
    }

    gamma_C
  }

  b <- boot(
    data = X,
    statistic = boot_fun,
    R = R,
    parallel = "multicore"
  )

  gamma_bar <- colMeans(b$t)
  gamma_mcse <- apply(b$t, 2, sd) / sqrt(R)

  mean_gamma <- mean(gamma_bar)
  mean_gamma_mcse <- sd(rowMeans(b$t)) / sqrt(R)

  list(
    gamma_bar = gamma_bar,
    gamma_mcse = gamma_mcse,
    mean_gamma = mean_gamma,
    mean_gamma_mcse = mean_gamma_mcse,
    boot_mat = b$t,
    boot_obj = b
  )
}

k.grid <- 3:7

ward.summary <- function(X, iters = 500, seed = 666) {
  ward_ch_summary <- ward_ch(X, k.grid)
  print(ward_ch_summary)
  ward_stab <- lapply(k.grid, function(k) {
    ward_jaccard_boot(X, k = k, R = iters, seed = seed)
  })
  names(ward_stab) <- paste0("k", k.grid)

  stab_summary <- data.frame(
    k = k.grid,
    mean_gamma = sapply(ward_stab, \(z) z$mean_gamma),
    mean_gamma_mcse = sapply(ward_stab, \(z) z$mean_gamma_mcse)
  )
  print(stab_summary)
}

ward.summary(X.pca[, 1:13], 500)
# k         CH
# 1 3 157.71526
# 2 4 128.48701
# 3 5 112.32357
# 4 6 103.55704
# 5 7  94.68403
#    k mean_gamma mean_gamma_mcse
# k3 3  0.6076719     0.003099954
# k4 4  0.5093733     0.002553549
# k5 5  0.4772990     0.003997354
# k6 6  0.4865560     0.004196353
# k7 7  0.4591030     0.003292549
```

5.7 Power Score Calculations

```
source("kmeans.R")

X <- X.pca[,1:13]

power_score <- function(datasets, label_col = "label", n_folds = 10, seed = 666,
    Bstart = 100) {
  set.seed(seed)
  ks <- names(datasets)

  results <- data.frame(
    k = as.numeric(gsub("k", "", ks)),
    PS = NA
  )

  power_score <- function(X_train, X_test, y_test, k, Bstart) {
    km_fit <- kmeans(X_train, centers = k, nstart = Bstart)
    centroids <- km_fit$centers
    train_clusters <- km_fit$cluster

    predict_clusters <- function(x) {
      dists <- apply(centroids, 1, function(centroid) sum((x - centroid)^2))
      which.min(dists)
    }

    test_clusters <- apply(X_test, 1, predict_clusters)

    ps_per_cluster <- numeric(k)
    for (j in 1:k) {
      idx <- which(test_clusters == j)
```

```r
      if (length(idx) < 2) {
        ps_per_cluster[j] <- 0
      } else {
        D <- outer(as.numeric(y_test[idx]), as.numeric(y_test[idx]), FUN =
            function(a, b) as.numeric(a == b))
        ps_per_cluster[j] <- sum(D) / (length(idx) * (length(idx) - 1))
      }
    }

    PS <- min(ps_per_cluster)
    return(PS)
  }

  for (ki in seq_along(ks)) {
    data <- datasets[[ki]]
    y <- factor(data[[label_col]])
    X <- data[, setdiff(names(data), label_col)]
    n <- nrow(X)
    folds <- sample(rep(1:n_folds, length.out = n))

    ps_folds <- numeric(n_folds)

    for (fold in 1:n_folds) {
      train_idx <- which(folds != fold)
      test_idx  <- which(folds == fold)

      X_train <- X[train_idx, ]
      X_test  <- X[test_idx, ]
      y_train <- y[train_idx]
      y_test  <- y[test_idx]

      k <- results$k[ki]
      ps_folds[fold] <- power_score(X_train, X_test, y_test, k, Bstart)
    }

    results$PS[ki] <- mean(ps_folds)
  }

  return(results)
}

datasets <- list(
  k3 = data_k3,
  k4 = data_k4,
  k5 = data_k5,
  k6 = data_k6,
  k7 = data_k7
)

results_power_score <- power_score(datasets, label_col = "label", n_folds = 10)
results_power_score
```

5.8 Misclassification Rates

```r
library(randomForest)
library(nnet)
library(xgboost)
library(tidyverse)
```

```r

supervised_cv_all <- function(datasets, label_col = "label", n_folds = 10, seed =
    666) {
  set.seed(seed)
  ks <- names(datasets)

  results <- data.frame(
    k = as.numeric(gsub("k", "", ks)),
    Random_Forest = NA,
    Multinomial_Logistic = NA,
    XGBoost = NA
  )

  for (ki in seq_along(ks)) {
    data <- datasets[[ki]]
    y <- factor(data[[label_col]])
    X <- data[, setdiff(names(data), label_col)]
    n <- nrow(X)
    folds <- sample(rep(1:n_folds, length.out = n))

    rf_mis <- numeric(n_folds)
    multinom_mis <- numeric(n_folds)
    xgb_mis <- numeric(n_folds)

    for (fold in 1:n_folds) {
      train_idx <- which(folds != fold)
      test_idx  <- which(folds == fold)

      X_train <- X[train_idx, ]
      X_test  <- X[test_idx, ]
      y_train <- y[train_idx]
      y_test  <- y[test_idx]

      # Random Forest
      rf_model <- randomForest(x = X_train, y = y_train, ntree = 500)
      rf_pred <- predict(rf_model, X_test)
      rf_mis[fold] <- mean(rf_pred != y_test)

      # Multinomial Logistic Regression
      multinom_model <- multinom(y_train ~ ., data = X_train, trace = FALSE)
      multinom_pred <- predict(multinom_model, X_test, type = "class")
      multinom_mis[fold] <- mean(multinom_pred != y_test)

      # XGBoost
      y_train_num <- as.numeric(y_train) - 1
      y_test_num <- as.numeric(y_test) - 1
      dtrain <- xgb.DMatrix(data = as.matrix(X_train), label = y_train_num)
      dtest  <- xgb.DMatrix(data = as.matrix(X_test), label = y_test_num)
      params <- list(objective = "multi:softmax", num_class = length(unique(y)),
          eval_metric = "merror")
      xgb_model <- xgb.train(params = params, data = dtrain, nrounds = 100,
          verbose = 0)
      xgb_pred <- predict(xgb_model, dtest)
      xgb_mis[fold] <- mean(xgb_pred != y_test_num)
    }

    results$Random_Forest[ki] <- mean(rf_mis)
    results$Multinomial_Logistic[ki] <- mean(multinom_mis)
    results$XGBoost[ki] <- mean(xgb_mis)
  }

  return(results)
}

datasets <- list(
  k3 = data_k3,
  k4 = data_k4,
  k5 = data_k5,
  k6 = data_k6,
  k7 = data_k7
)

results_all <- supervised_cv_all(datasets, label_col = "label", n_folds = 10)
results_all



supervised_cv_confidence <- function(datasets, label_col = "label", n_folds = 10,
    seed = 666) {
  set.seed(seed)
  ks <- names(datasets)

  results <- data.frame(
    k = as.numeric(gsub("k", "", ks)),
    Random_Forest = NA,
    Multinomial_Logistic = NA,
    XGBoost = NA
  )

  for (ki in seq_along(ks)) {
    data <- datasets[[ki]]
    y <- factor(data[[label_col]])
    X <- data[, setdiff(names(data), label_col)]
    n <- nrow(X)
    folds <- sample(rep(1:n_folds, length.out = n))

    rf_conf <- numeric(n_folds)
    multinom_conf <- numeric(n_folds)
    xgb_conf <- numeric(n_folds)

    for (fold in 1:n_folds) {
      train_idx <- which(folds != fold)
      test_idx  <- which(folds == fold)

      X_train <- X[train_idx, ]
      X_test  <- X[test_idx, ]
      y_train <- y[train_idx]
      y_test  <- y[test_idx]

      # Random Forest
      rf_model <- randomForest(x = X_train, y = y_train, ntree = 500)
      rf_pred_prob <- predict(rf_model, X_test, type = "prob")
      rf_conf[fold] <- mean(apply(rf_pred_prob, 1, max))

      # Multinomial Logistic Regression
      multinom_model <- nnet::multinom(y_train ~ ., data = X_train, trace = FALSE)
      multinom_pred_prob <- predict(multinom_model, X_test, type = "probs")
      multinom_conf[fold] <- mean(apply(multinom_pred_prob, 1, max))

      # XGBoost
      y_train_num <- as.numeric(y_train) - 1
      y_test_num <- as.numeric(y_test) - 1
      dtrain <- xgb.DMatrix(data = as.matrix(X_train), label = y_train_num)
      dtest  <- xgb.DMatrix(data = as.matrix(X_test), label = y_test_num)
      params <- list(objective = "multi:softprob", num_class = length(unique(y)), eval_metric = "mlogloss")
      xgb_model <- xgb.train(params = params, data = dtrain, nrounds = 100, verbose = 0)
      xgb_pred_prob <- matrix(predict(xgb_model, dtest), ncol = length(unique(y)), byrow = TRUE)
      xgb_conf[fold] <- mean(apply(xgb_pred_prob, 1, max))
    }

    results$Random_Forest[ki] <- mean(rf_conf)
    results$Multinomial_Logistic[ki] <- mean(multinom_conf)
    results$XGBoost[ki] <- mean(xgb_conf)
  }

  return(results)
}

# Example usage
results_all_conf <- supervised_cv_confidence(datasets, label_col = "label", n_folds = 5)
results_all_conf
```

5.9 Heatmaps by Genre

```r

source("themes.R")
source("data.R")

library(tidyverse)

k3 <- read.csv("Data/dat-3.csv")

all_data <- bind_cols(k3, collins)

mean_mat <- all_data %>%
  group_by(label) %>%
  summarize(across(
    c(
      "FirstPerson",
      "InnerThinking",
      "ThinkPositive",
      "ThinkNegative",
      "ThinkAhead",
      "ThinkBack",
      "Reasoning",
      "Share_SocTies",
      "Direct_Activity",
      "Interacting",
      "Notifying",
      "LinearGuidance",
      "WordPicture",
      "SpaceInterval",
      "Motion",
      "PastEvents",
      "TimeInterval",
      "ShiftingEvents",
      "Text_Coverage"
    ),
    mean
  )) %>%
  column_to_rownames("label") %>%
  as.matrix()

tbl <- table(all_data$Genre, all_data$label)
tbl_prop <- prop.table(tbl, margin = 2)
df_heat_prop <- as.data.frame(tbl_prop)
colnames(df_heat_prop) <- c("genre", "cluster", "proportion")
df_heat_prop <- df_heat_prop %>%
  mutate(
    genre_names = case_when(
      genre == 1 ~ "Press: Reporting",
      genre == 2 ~ "Press: Editorial",
      genre == 3 ~ "Press: Reviews",
      genre == 4 ~ "Religion",
      genre == 5 ~ "Skills and Hobbies",
      genre == 6 ~ "Popular Lore",
      genre == 7 ~ "Biography, Memoirs, Belles Lettres, etc.",
      genre == 8 ~ "Official Communications",
      genre == 9 ~ "Learned (scholarly journals)",
      genre == 10 ~ "General Fiction",
      genre == 11 ~ "Mystery and Detective Fiction",
      genre == 12 ~ "Science Fiction",
      genre == 13 ~ "Adventure and Western Fiction",
      genre == 14 ~ "Romance and Love Story",
      genre == 15 ~ "Humor"
    ),
    genre_names = factor(
      genre_names,
      levels = c(
        "Press: Reporting",
        "Press: Editorial",
        "Press: Reviews",
        "Religion",
        "Skills and Hobbies",
        "Popular Lore",
        "Biography, Memoirs, Belles Lettres, etc.",
        "Official Communications",
        "Learned (scholarly journals)",
        "General Fiction",
        "Mystery and Detective Fiction",
        "Science Fiction",
        "Adventure and Western Fiction",
        "Romance and Love Story",
        "Humor"
      )
    ),
    category = case_when(
      genre %in% c(1, 2, 3) ~ "Press",
      genre %in% c(4, 5, 6) ~ "Non-press Nonfiction",
      genre == 7 ~ "Biography",
      genre %in% c(8, 9) ~ "Scholarship and Official Documents",
      genre %in% c(10, 11, 12, 13, 14, 15) ~ "Fiction"
    ),
    category = factor(
      category,
      levels = c(
        "Press",
        "Non-press Nonfiction",
        "Biography",
        "Scholarship and Official Documents",
        "Fiction"
      )
    )
  )


# Heatmap by genre
p1 <- ggplot(
  df_heat_prop,
  aes(x = cluster, y = genre_names, fill = proportion)
) +
  geom_tile() +
  geom_text(aes(label = round(proportion, 2)), color = "black", size = 16) + #
      larger numbers on tiles
  scale_fill_gradient2_paper() +
  theme_paper(base_size = 64, text_size = 48) +
  labs(x = "Cluster", y = "Genre", fill = "Proportion") +
  theme(
    legend.position = "none",
    panel.grid = element_blank()
  )

k5 <- read.csv("Data/dat-5.csv")
all_data2 <- bind_cols(k5, collins)

all_data2$category <- case_when(
  all_data2$Genre %in% c(1, 2, 3) ~ "Press",
  all_data2$Genre %in% c(4, 5, 6) ~ "Non-press Nonfiction",
  all_data2$Genre == 7 ~ "Biography",
  all_data2$Genre %in% c(8, 9) ~ "Scholarship and Official Documents",
  all_data2$Genre %in% c(10, 11, 12, 13, 14, 15) ~ "Fiction"
)

tbl <- table(all_data2$category, all_data2$label)
tbl_prop <- prop.table(tbl, margin = 2)
df_heat_prop <- as.data.frame(tbl_prop)
colnames(df_heat_prop) <- c("category", "cluster", "proportion")

df_heat_prop_cat <- df_heat_prop %>%
  group_by(category, cluster) %>%
  summarize(proportion = sum(proportion)) %>%
  ungroup()

# Heatmap by category
p3 <- ggplot(
  df_heat_prop_cat,
  aes(x = cluster, y = category, fill = proportion)
) +
  geom_tile() +
  geom_text(aes(label = round(proportion, 2)), color = "black", size = 16) +
  scale_fill_gradient2_paper() +
  theme_paper(base_size = 64, text_size = 48) +
  labs(x = "Cluster", y = "Category", fill = "Proportion") +
  theme(
    legend.position = "none",
    panel.grid = element_blank()
  )
```

5.10 Linear Discriminant Analysis

```r
library(MASS)

source("data.R")

X$label <- as.factor(k3$label)
X_scaled <- scale(X[, -which(names(X) == "label")])
lda_model <- lda(X$label ~ ., data = as.data.frame(X_scaled))

lda_model$scaling
```

5.11 Figures

5.11.1 Theme for Paper

```r
##Themes.R

library(ggplot2)
library(sysfonts)

font_add("cm", regular = "fonts/cmunrm.ttf")
showtext::showtext_auto()

theme_paper <- function(base_size = 14, text_size = 14, base_family = "cm") {
  theme_minimal(base_size = base_size, base_family = base_family) %+replace%
    theme(
      # Text
      plot.title = element_text(
        face = "bold",
        size = text_size + 2,
        hjust = 0.5,
        margin = margin(b = 6)
      ),
      plot.subtitle = element_text(
        size = text_size,
        hjust = 0.5,
        margin = margin(b = 8)
      ),
      plot.caption = element_text(
        size = text_size - 2,
        hjust = 1,
        margin = margin(t = 6)
      ),
      axis.title.x = element_text(size = text_size, margin = margin(t = 8)),
      axis.title.y = element_text(
        size = text_size,
        angle = 90,
        margin = margin(r = 8)
      ),
      axis.text = element_text(size = text_size - 1),

      # Panel & grid
      panel.grid.major.x = element_blank(),
      panel.grid.minor.x = element_blank(),
      panel.grid.minor.y = element_blank(),
      #panel.grid.major.y = element_line(linewidth = 0.3, color = "grey85"),
      panel.border = element_rect(fill = NA, color = "grey60", linewidth = 0.4),

      # Legend
      legend.position = "bottom",
      legend.title = element_text(size = base_size - 1, face = "bold"),
      legend.text = element_text(size = base_size - 2),
      legend.key.width = unit(1.2, "lines"),

      # Strips (for facets)
      strip.background = element_rect(fill = "grey90", color = NA),
      strip.text = element_text(face = "bold", size = base_size - 1),

      # Margins
      plot.margin = margin(t = 8, r = 8, b = 8, l = 8)
    )
}

paper_palette <- c(
  "#1b9e77",
  "#d95f02",
  "#7570b3",
  "#e7298a",
  "#66a61e",
  "#e6ab02"
)

scale_color_paper <- function(...) {
  scale_color_manual(values = paper_palette, ...)
}

scale_fill_paper <- function(...) {
  scale_fill_manual(values = paper_palette, ...)
}

scale_fill_gradient2_paper <- function(
  low = "#7570b3",
  mid = "white",
  high = "#d95f02",
  midpoint = 0,
  ...
) {
  scale_fill_gradient2(
    low = low,
    mid = mid,
    high = high,
    midpoint = midpoint,
    ...
  )
}
```

5.11.2 Reproduce Figures

```r
source("themes.R")
source("data.R")

library(patchwork)

#PCA ########################################################

pca_elbow_plot(X, center = T, scale. = T)
ggsave("Figures/PCA-elbow.png", width = 6, height = 5, dpi = 600)

#How many components should we use according to a scree plot?
max(which(eigen(cor(X))$values >= 1))

#KMEANS #####################################################

source("kmeans.R")

kg <- kmeans_clusters(X.pca[, 1:13], 3, Bstart = 10)
kg.df <- data.frame(
  pc1 = X.pca[, 1],
  pc2 = X.pca[, 2],
  labels = factor(kg)
)

kmeans.plot <- ggplot(kg.df, aes(x = pc1, y = pc2, color = labels)) +
  geom_point(size = 0.5) +
  scale_color_paper(guide = "none") +
  labs(
    x = expression(PC[1]),
    y = expression(PC[2])
  ) +
  theme_paper(text_size = 64)
ggsave("Figures/kmeans.png", kmeans.plot, width = 5, height = 4, dpi = 600)

# Genre Classification #######################################

source("cluster_genres.R")

genre.cat <- p1 + p3

ggsave(
  "Figures/cluster_genre_heatmaps.png",
  genre.cat,
  width = 10,
  height = 4,
  dpi = 600
)
```

6 Appendix - SAS Code

6.1 MANOVA

```sas

/* Code generated on: 11/14/25, 1:43 PM */

/****************************************
```

```
                  K = 3
*****************************************/

FILENAME REFFILE '/home/u64365443/sasuser.v94/dat-3.csv';

PROC IMPORT DATAFILE=REFFILE
        DBMS=CSV
        OUT=WORK.PCA;
        GETNAMES=YES;
RUN;


ods select CharStruct MultStat;

proc glm data = PCA;
    class label;
    model PC1 PC2 PC3 PC4 PC5 PC6 PC7 PC8 PC9 PC10 PC11 PC12 PC13 =
          label;

    /* Request Wilks', Pillai's, Hotelling-Lawley, Roy's tests */
    manova h = label / printe printh;
run;
quit;

/*****************************************
                  K = 4
*****************************************/

%web_drop_table(WORK.PCA);

FILENAME REFFILE '/home/u64365443/sasuser.v94/dat-4.csv';

PROC IMPORT DATAFILE=REFFILE
        DBMS=CSV
        OUT=WORK.PCA;
        GETNAMES=YES;
RUN;

%web_open_table(WORK.PCA);

ods select CharStruct MultStat;
proc glm data = PCA;
    class label;
    model PC1 PC2 PC3 PC4 PC5 PC6 PC7 PC8 PC9 PC10 PC11 PC12 PC13 =
          label;

    /* Request Wilks', Pillai's, Hotelling-Lawley, Roy's tests */
    manova h = label / printe printh;
run;
quit;

/*****************************************
                  K = 5
*****************************************/

%web_drop_table(WORK.PCA);

FILENAME REFFILE '/home/u64365443/sasuser.v94/dat-5.csv';

```

```
PROC IMPORT DATAFILE=REFFILE
        DBMS=CSV
        OUT=WORK.PCA;
        GETNAMES=YES;
RUN;

%web_open_table(WORK.PCA);

ods select CharStruct MultStat;
proc glm data = PCA;
    class label;
    model PC1 PC2 PC3 PC4 PC5 PC6 PC7 PC8 PC9 PC10 PC11 PC12 PC13 =
          label;

    /* Request Wilks', Pillai's, Hotelling-Lawley, Roy's tests */
    manova h = label / printe printh;
run;
quit;

/*****************************************
                  K = 6
*****************************************/

%web_drop_table(WORK.PCA);

FILENAME REFFILE '/home/u64365443/sasuser.v94/dat-6.csv';

PROC IMPORT DATAFILE=REFFILE
        DBMS=CSV
        OUT=WORK.PCA;
        GETNAMES=YES;
RUN;

%web_open_table(WORK.PCA);

ods select CharStruct MultStat;
proc glm data = PCA;
    class label;
    model PC1 PC2 PC3 PC4 PC5 PC6 PC7 PC8 PC9 PC10 PC11 PC12 PC13 =
          label;

    /* Request Wilks', Pillai's, Hotelling-Lawley, Roy's tests */
    manova h = label / printe printh;
run;
quit;

/*****************************************
                  K = 7
*****************************************/

%web_drop_table(WORK.PCA);

FILENAME REFFILE '/home/u64365443/sasuser.v94/dat-7.csv';

PROC IMPORT DATAFILE=REFFILE
        DBMS=CSV
        OUT=WORK.PCA;
        GETNAMES=YES;
RUN;
```

```
%web_open_table(WORK.PCA);

ods select CharStruct MultStat;
proc glm data = PCA;
    class label;
    model PC1 PC2 PC3 PC4 PC5 PC6 PC7 PC8 PC9 PC10 PC11 PC12 PC13 =
          label;

    /* Request Wilks', Pillai's, Hotelling-Lawley, Roy's tests */
    manova h = label / printe printh;
run;
quit;
```