Empirical Benchmarking of Python and R for Sentiment Classification and Topic Modeling: A Criteria-Based Decision Framework

Ezozbek Tokhirov[1], Odilzhan Turdiev[1], Tolaniddin Nurmukhamedov[1], Mukhabbat Kurbanova[1] and Mohammed Sami[2]

[1]*Department of Information Systems and Technologies in Transport, Tashkent State Transport University, 100167 Tashkent, Uzbekistan*

[2]*University of Diyala, Baghdad Old Road, 32001 Diyala, Iraq*

ezzoozz@gmail.com, odiljan.turdiev@mail.ru, ntolaniddin@mail.ru, kmukhabbat2507@gmail.com, dr.mohammed.sami@uodiyala.edu.iq

Keywords: Python, R, Sentiment Classification, Topic Modeling, Empirical Benchmark, Cross-Validation, WebAssembly, Data Analytics.

Abstract: This study presents a controlled empirical benchmark of Python and R for sentiment classification and topic modeling, addressing the ongoing debate regarding the relative suitability of both programming environments for natural language processing tasks. To ensure a fair comparison, identical experimental conditions were established for both platforms. Using the Pang-Lee corpus of 10,662 labeled review snippets, mirrored multinomial Naive Bayes implementations were evaluated by stratified five-fold cross-validation, while matched six-topic Latent Dirichlet Allocation (LDA) models employed identical preprocessing procedures, tokenization, vocabulary construction, hyperparameter settings, initialization strategy, and Gibbs-sampling order. Classification performance was assessed using Accuracy, Precision, Recall, F1-score, and ROC-AUC, whereas topic-model quality was evaluated using UMass coherence and perplexity. Execution performance was measured under identical WebAssembly-based environments to eliminate hardware-related bias. The experimental results demonstrate that Python and R produced identical predictive performance, achieving a mean Accuracy of 0.7459, F1-score of 0.7450, ROC-AUC of 0.8248, UMass coherence of −2.9997, and perplexity of 318.7501. Although model quality remained unchanged across programming languages, substantial differences were observed in execution efficiency, with the median complete-pipeline runtime equal to 0.833 s for Python and 17.220 s for R under the same execution conditions. Based on these findings, a seven-criterion decision framework is proposed that distinguishes predictive performance, topic-model quality, execution efficiency, statistical workflow, visualization capabilities, reproducibility, and deployment considerations. The proposed framework provides practical guidance for researchers and practitioners selecting between Python, R, or hybrid analytical workflows for text mining and natural language processing applications.

1 INTRODUCTION

Python and R are widely used for data analysis but emerged from different programming traditions. Python is a general-purpose language with broad software-engineering and machine-learning applications [1], while pandas established mature tabular-analysis workflows [2]. R and RStudio are closely associated with statistical computing [3], and R visualization has a strong publication-oriented ecosystem [4]. Applied engineering research increasingly converts operational, spatial, and monitoring data into computational decision evidence, including biomechanical modeling [5], data-driven risk prioritization [6], 3D reservoir modeling [7], and geospatial infrastructure monitoring [8].

These differences are especially visible in quantitative and text-oriented analysis. R provides specialized statistical interfaces, including mature econometric tooling [9]. Python offers long-established natural-language-processing resources such as NLTK [10] and production-oriented NLP

pipelines such as spaCy [11]. R also provides dedicated text-analysis infrastructure through quanteda [12] and earlier text-mining packages [13].

Workflow support is another practical difference. R Markdown combines code, narrative, and results [14], while Jupyter notebooks provide a related computational-document model for Python [15]. Cross-language embedding is also feasible [16], so mixed workflows can integrate both environments.

Many Python-versus-R comparisons remain narrative: they list packages or syntax preferences without testing mirrored algorithms on identical observations, folds, random-number sequences, and hardware. Consequently, differences in model quality may originate from preprocessing, initialization, or hyperparameters rather than the language itself, while identical numerical quality may still coexist with runtime and workflow differences.

This study addresses that gap through: (1) stratified 5-fold MNB sentiment classification with identical features and folds; (2) matched LDA using the same Gibbs-sampling sequence; and (3) a seven-criterion framework that separates measured empirical criteria from workflow criteria supported by the literature.

The study asks three research questions: RQ1) Do mirrored Python and R sentiment-classification implementations produce materially different predictive metrics? RQ2) Do matched Python and R LDA implementations produce materially different topic-model quality metrics? RQ3) How do measured execution time and workflow-oriented criteria affect the choice between the two environments?

The benchmark was evaluated using the following equivalence hypotheses:

- H1. Under identical data, vocabulary, folds, model equations, and smoothing, the absolute difference between the mean Python and R classification metrics will be below 0.001.
- H2. Under identical token sequences, LDA hyperparameters, pseudo-random number generator, initialization, and update order, Python and R will produce the same coherence and perplexity values to the reported numerical precision.
- H3. Even when model-quality metrics are equivalent, runtime and workflow criteria will differ; consequently, neither language will dominate all seven criteria of the proposed framework.

2 LITERATURE REVIEW

R was designed within the statistical-computing tradition and is distinguished by formula-based model specification and specialized packages; plm is one example of econometric support [9]. RStudio provides an integrated environment [3], while publication-oriented visualization is supported by the R graphics ecosystem [4]. Python follows a general-purpose programming model [1], with pandas providing explicit DataFrame-oriented manipulation [2]. These differences motivate workflow criteria but do not establish superiority in predictive quality.

Text analytics further illustrates the need to distinguish algorithms from ecosystems. NLTK and spaCy provide broad and production-oriented NLP resources [10], [11], while quanteda and earlier R text-mining infrastructure support quantitative text analysis [12], [13]. LDA is a probabilistic generative model independent of implementation language [17]; a meaningful comparison should therefore control equations and stochastic sampling.

Reproducibility is equally important. R Markdown supports executable reporting [14], while Jupyter notebooks provide a computational publishing format [15]. In the present benchmark, reproducibility is strengthened by a public labeled corpus [18] and matched WebAssembly runtimes: Pyodide runs CPython through WebAssembly [19], whereas webR compiles the R interpreter for browser and Node.js execution [20].

3 METHODOLOGY

The methodology follows four phases. Phase 1 fixes the corpus, tokenization, vocabularies, and fold assignments. Phase 2 runs mirrored multinomial Naive Bayes (MNB) with 5-fold cross-validation. Phase 3 runs matched collapsed-Gibbs LDA with a shared pseudo-random number generator and sampling order. Phase 4 measures repeated wall-clock time and combines empirical outcomes with workflow criteria.

No Accuracy, F1, ROC-AUC, coherence, perplexity, or execution-time value in the Results section was inferred from documentation or inserted as an illustrative value. All reported values were produced by executing the benchmark scripts.

Table 1: Operational criteria used in the Python-R decision framework.

Criterion	Operationalization
Predictive quality	5-fold Accuracy, Precision, Recall, F1, and ROC-AUC; mean ± SD.
Topic-model quality	UMass coherence and model-fit perplexity under matched LDA.
Execution time	Median wall-clock time after one warm-up and five measured repetitions.
Statistical workflow	Formula-oriented model specification and diagnostic/statistical tooling.
Visualization	Publication-oriented composition and extensibility of graphics workflows.
Reproducibility	Explicit environments, fixed exchange files, and executable reporting support.
Deployment	Ease of embedding analyses in applications, services, or shareable interfaces.

The timed code used language-native data structures and arithmetic, while both implementations followed the same equations and consumed the same preprocessed exchange files.

3.1 Research Design

The research design is a controlled, task-based cross-language benchmark. Predictive and topic-model quality are treated as algorithmic outcomes; execution time is treated as a runtime-specific outcome; and statistical workflow, visualization, reproducibility, and deployment are treated as workflow-level criteria. Table 1 defines the seven criteria. The first three are directly measured in the present experiments, while the remaining four are interpreted using the documented capabilities and literature reviewed in Section 2.

3.2 Data

The experiments use sentence polarity dataset v1.0 introduced by Pang and Lee [18]. The corpus contains 10,662 processed movie-review snippets: 5,331 positive and 5,331 negative observations. The source files provide one lowercased snippet per line, with labels originating from the corpus polarity annotations. All observations were retained for classification. For LDA, a deterministic balanced subset of 2,000 snippets was selected; after requiring at least three retained content tokens, 1,671 documents and 10,398 tokens remained.

3.3 Preprocessing and Shared Tasks

The same preprocessing exchange files were supplied to both languages. No language-specific tokenizer, feature selector, or stop-word package was used in the measured benchmark.

Tokenization was performed using the lowercase regular expression [a-z]+(?:'[a-z]+)?. For classification, the 2,000 most frequent label-blind unigrams formed a fixed shared vocabulary. For LDA, a fixed stop-word list was applied, the 800 most frequent remaining terms formed the vocabulary, and each document was truncated to at most 50 retained tokens. Deterministic stratified folds and the LDA subset were generated with the same explicitly specified 32-bit linear congruential generator (LCG).

3.4 Matched Execution Environment

Both language runtimes were executed sequentially on the same virtualized x86-64 Debian GNU/Linux 13 host under Node.js 22.16.0. The container exposed 56 logical processors and approximately 4.0 GiB of RAM; the benchmark code itself was single-threaded. Python 3.14.2 was executed through Pyodide 314.0.2, a CPython WebAssembly distribution [19].

R 4.6.0 was executed through webR 0.6.0, which provides an R interpreter compiled to WebAssembly for browser and Node.js environments [20]. Using a common host and WebAssembly execution model removes between-machine hardware variation. However, the measured timing values characterize these specific runtime implementations and must not be interpreted as universal native Python-versus-native R speed rankings.

3.5 Sentiment Classification Benchmark

Sentiment polarity was modeled with multinomial Naive Bayes because the same model can be specified directly from count data in both languages without package-specific training behavior. For class c and term j, Laplace smoothing alpha = 1 was used. The class score for document x was computed from the log class prior and the sum of term counts multiplied by smoothed class-conditional log probabilities.

The MNB decision rule was implemented as:

$$score(c|x) = \log P(c) + \Sigma_j x_j \log[(N_{cj} + \alpha)/(N_c + \alpha V)]. \quad (1)$$

where $V = 2{,}000$, N_{cj} is the training count of term j in class c, and N_c is the total number of retained training tokens in class c.

The positive-class score difference score(1|x) - score(0|x) was used for ROC-AUC ranking; a non-negative difference produced the positive class prediction. Thus, Python and R used the same features, class priors, smoothing, and decision threshold.

Five folds were assigned independently within each class by deterministic LCG shuffling and cyclic fold labels 1-5. Each fold served once as the test fold. Accuracy, precision, recall, F1-score, and ROC-AUC were calculated for every fold and summarized as mean ± sample standard deviation.

To control cross-language equivalence, fold identifiers were generated before either language was executed and stored in the shared classification file. A single label-blind vocabulary was fixed before cross-validation and reused in both implementations; therefore, the reported classification scores are comparative benchmark values rather than estimates from a nested feature-selection pipeline.

The same sparse term counts were read by both scripts. Neither implementation called a machine-learning package inside the model-fitting region. This choice intentionally isolates the language/runtime execution of the same MNB procedure rather than comparing different library defaults.

H1 was evaluated using the absolute difference between the Python and R mean values for each of the five predictive metrics. The operational equivalence threshold used in this study was 0.001.

The ROC-AUC calculation used rank statistics with average ranks for tied scores in both implementations. This avoids differences caused by package-specific interpolation or curve-integration conventions.

The classification benchmark therefore tests numerical and implementation equivalence under a mirrored protocol, not the superiority of one classifier family over another.

Reproducibility was reinforced by using the same plain-text sparse document representation, term identifiers, fold labels, LDA token sequences, LCG constants, and benchmark order in both environments. Runtime versions and all model hyperparameters are stated explicitly.

3.6 LDA Topic-Model Benchmark

A six-topic LDA model was fitted to the 1,671-document text subset. The implementation follows the generative LDA formulation [17] and uses collapsed Gibbs sampling. The number of topics was K = 6, alpha = 0.1, beta = 0.01, and 20 Gibbs iterations. Topic assignments were initialized with LCG seed 7301 and updated in identical document and token order.

$$P(z_i = k \mid rest) \propto (n_{dk} + \alpha)(n_{kw} + \beta)/(n_k + V\beta). \quad (2)$$

Here ndk is the document-topic count, nkw is the topic-term count, and nk is the total number of tokens assigned to topic k; all counts in (2) are evaluated after removing the current token assignment. The same 32-bit LCG used constants 1,664,525 and 1,013,904,223 modulo 2^32. This made initialization and every categorical sampling draw reproducible across Python and R.

$$Perplexity = \exp\{-(1/N)\, \Sigma_i \log[\Sigma_k\, \theta_{dk}\, \varphi_{kwi}]\}. \quad (3)$$

Posterior-mean theta and phi estimates were used to calculate model-fit perplexity. Topic interpretability was summarized by UMass coherence over the top 10 terms of each topic. Higher (less negative) UMass coherence and lower perplexity indicate better values under the respective metrics.

3.7 Timing and Evaluation Protocol

Classification and LDA fitting-plus-scoring were timed separately. File parsing and runtime initialization were excluded from the timed regions. Each task was executed once as an untimed warm-up and then five times. The median of the five measured wall-clock times was reported to reduce the influence of transient runtime variation. The complete-pipeline time is the sum of the two task medians.

$$C_{UMass} = mean\{\log[(D(w_m, w_l) + 1)/D(w_l)]\}. \quad (4)$$

$D(w_l)$ is the number of documents containing term w_l and $D(w_m, w_l)$ is the number containing both terms. H2 was evaluated by comparing the two coherence and perplexity outputs to the reported numerical precision. H3 was evaluated across the full seven-criterion framework.

The benchmark does not treat a WebAssembly timing advantage as a universal property of a programming language. Instead, runtime is reported as one measured criterion for the specified environment. Statistical workflow, visualization, reproducibility, and deployment are assessed separately so that algorithmic quality is not conflated with ecosystem characteristics.

The final decision rule is criterion based: a language is favored only for a criterion supported by either a measured difference in this benchmark or a clearly defined workflow capability supported by the literature. Ties are retained when the evidence does not justify assigning an advantage.

Table 2: Measured empirical benchmark results for the matched Python and R implementations.

Benchmark	Metric	R	Python
Classification	Accuracy	0.7459 ± 0.0068	0.7459 ± 0.0068
Classification	Precision	0.7475 ± 0.0039	0.7475 ± 0.0039
Classification	Recall	0.7426 ± 0.0176	0.7426 ± 0.0176
Classification	F1-score	0.7450 ± 0.0094	0.7450 ± 0.0094
Classification	ROC-AUC	0.8248 ± 0.0082	0.8248 ± 0.0082
LDA	UMass coherence	-2.999655	-2.999655
LDA	Perplexity	318.7501	318.7501
Timing	5-fold classification, median s	4.133	0.109
Timing	LDA fitting + scoring, median s	13.087	0.723
Timing	Complete empirical pipeline, median s	17.220	0.833

Note: classification values are mean ± sample SD across five folds. Coherence and perplexity are deterministic outputs of the matched LDA run. Timing values are medians of five measured repetitions after one warm-up and characterize only the specified Pyodide/webR WebAssembly configuration.

4 RESULTS

Table 2 reports the five cross-validated classification metrics, two LDA quality metrics, and three execution-time measurements. Python and R were identical on model-quality values, while runtime differed.

4.1 Classification Results

Across the five stratified test folds, both implementations achieved mean accuracy 0.7459 ± 0.0068, precision 0.7475 ± 0.0039, recall 0.7426 ± 0.0176, F1-score 0.7450 ± 0.0094, and ROC-AUC 0.8248 ± 0.0082. The absolute Python-R difference in the mean of every metric was 0.0000 to the reported precision. Fold-level values were also identical, demonstrating that the shared counts, priors, smoothing, ranking rule, and decision threshold were reproduced consistently.

4.2 Topic-Model Results

The matched LDA implementations produced UMass coherence -2.999655 and perplexity 318.750080 in both languages. Topic assignments and top-10 term identifiers for all six topics were also identical, supporting the numerical-equivalence objective of the matched protocol.

4.3 Execution-Time Results

The median 5-fold classification time was 4.133 s in R and 0.109 s in Python, a 37.92-fold ratio. LDA fitting and scoring required 13.087 s in R and 0.723 s in Python, an 18.09-fold ratio. The complete empirical pipeline required 17.220 s and 0.833 s, respectively. The R/Python complete-pipeline time ratio was 20.68 in the tested matched WebAssembly configuration. This result is environment specific and is not generalized to native installations.

4.4 Hypothesis Evaluation

H1 was supported: the absolute difference between Python and R was below 0.001 for Accuracy, Precision, Recall, F1, and ROC-AUC; all mean differences were 0.0000 at the reported precision. H2 was supported: coherence and perplexity were identical to the reported precision and the top-term identifiers matched. H3 was also supported: model quality was tied, but measured runtime favored Python in this environment, statistical and visualization workflow criteria favored R, reproducibility was treated as a tie, and deployment favored Python. No language dominated all seven criteria.

4.5 Evidence-Based Decision Framework

The resulting framework assigns: predictive quality - Tie; topic-model quality - Tie; execution time under the tested WebAssembly configuration - Python; statistical workflow - R; visualization - R; reproducibility - Tie; and deployment - Python. The framework therefore recommends selecting by task boundary rather than by a universal language ranking. For an analysis whose primary requirement is the mirrored model quality tested here, either language is acceptable. For WebAssembly execution of these pure-loop implementations, Python was faster. For formula-oriented statistical work and publication graphics, the R ecosystem remains advantageous [3], [4], [9]. For NLP integration and software-system

embedding, Python has broader relevant infrastructure [1], [11], [16].

5 DISCUSSION

The central empirical finding is that programming language alone did not change model quality when the mathematical procedure and all stochastic choices were controlled. The identical cross-validation metrics are not a claim that Python and R are universally equivalent; rather, they show that the same MNB count model, applied to the same features and folds, produces the same predictive evidence. This distinction matters because ecosystem comparisons often attribute algorithmic outcomes to languages even when different libraries, preprocessors, or defaults were used.

The observed Accuracy and F1 values describe a deliberately simple unigram MNB baseline on the Pang-Lee sentence-polarity corpus [18]. The purpose of the classifier was not to establish a new state of the art in sentiment analysis. NLTK [10] and spaCy [11] illustrate richer Python NLP pipelines, while quanteda [12] and R text-mining infrastructure [13] offer extensive alternatives. The contribution is the controlled cross-language comparison, not a new sentiment architecture.

The LDA experiment leads to the same methodological conclusion. LDA is defined independently of the implementation language [17]. By fixing the token sequence, K, alpha, beta, initialization, LCG, and update order, both implementations generated identical topic-quality values and top-term identifiers. Therefore, claims such as “Python produces better topics” or “R produces more interpretable topics” cannot be inferred from language choice alone without controlled implementation-level comparison.

Execution time did differ strongly. In the tested WebAssembly configuration, the Python implementation required approximately 1/20.68 of the R runtime for the combined measured pipeline. However, Pyodide [19] and webR [20] are distinct WebAssembly runtime projects, and pure interpreted loops may incur different implementation costs. The result is therefore a valid measurement of the specified configuration but not a native-language benchmark. This limitation is explicitly retained in the decision framework rather than hidden behind a general scalability claim.

Workflow criteria remain relevant after algorithmic equivalence is established. R provides formula-oriented statistical conventions and specialized methods [9], an integrated statistical working environment [3], and publication-oriented graphics [4]. Python benefits from a general-purpose software model [1], mature tabular processing [2], established NLP libraries [10], [11], and embedding in larger software systems [16]. These are ecosystem and engineering considerations, not substitutes for measured predictive metrics.

Reproducibility is a genuine tie in the framework. R Markdown can connect code, text, and output in a single report [14], and Jupyter notebooks support a comparable computational-document model [15]. The present benchmark adds another reproducibility layer by externalizing folds and token identifiers before either language is run. This prevents hidden differences in random splitting and preprocessing, two common sources of irreproducible cross-language comparisons.

The study has four limitations: one public sentence-polarity corpus, a baseline MNB classifier, two LDA diagnostics, and WebAssembly rather than native runtimes. These constraints narrow external validity, particularly for timing, but do not invalidate the measured numerical equivalence under the controlled protocol.

6 CONCLUSIONS

This study converts a descriptive Python-versus-R comparison into a measured empirical benchmark. The Pang-Lee corpus of 10,662 labeled review snippets was analyzed with mirrored 5-fold MNB classification and matched six-topic LDA, while data, vocabulary, folds, equations, initialization, and sampling order were controlled.

Python and R produced identical classification means: Accuracy 0.7459, Precision 0.7475, Recall 0.7426, F1 0.7450, and ROC-AUC 0.8248. LDA quality was likewise identical, with UMass coherence -2.9997 and perplexity 318.7501. These results support H1 and H2 and show that mirrored algorithms did not change model quality across languages.

Runtime differed: the complete measured pipeline required 0.833 s in Python and 17.220 s in R, a 20.68-fold difference in the tested Pyodide/webR configuration. H3 was also supported because no language dominated all seven criteria: model quality was tied, measured WebAssembly runtime and deployment favored Python, statistical workflow and visualization favored R, and reproducibility was tied.

The main contribution is the separation of algorithmic quality from runtime and ecosystem effects through cross-validated metrics, topic coherence, perplexity, and repeated execution times. Future work should repeat the protocol on engineering-text datasets and native x86-64 installations, include alternative classifiers and

transformer models, evaluate additional topic metrics, and measure peak memory.

REFERENCES

[1] P. Joyce, "Python Programming," in C and Python Applications, pp. 1-57, 2021, [Online]. Available: https://doi.org/10.1007/978-1-4842-7774-4_1.

[2] W. McKinney, Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython. O'Reilly Media, 2012, [Online]. Available: https://doi.org/10.5555/2361993.

[3] K. I. Musa, W. N. A. W. Mansor, and T. M. Hanis, "R, RStudio and RStudio Cloud," in Data Analysis in Medicine and Health Using R, pp. 1-14, 2023, [Online]. Available: https://doi.org/10.1201/9781003296775-1.

[4] C. Sievert, Interactive Web-Based Data Visualization with R, plotly, and shiny. Chapman and Hall/CRC, 2020, [Online]. Available: https://doi.org/10.1201/9780429447273.

[5] M. M. Talipov, "Computational modeling and analysis of mechanical power consumption in train assemblers' work," Proceedings of International Conference on Applied Innovation in IT, vol. 13, no. 2, pp. 419-426, 2025, [Online]. Available: https://doi.org/10.25673/120513.

[6] R. Salmorbekova and M. Talipov, "Digital risk matrix and safety management workflow for airport infrastructure in developing countries: a data-driven prioritization approach," Vibroengineering Procedia, vol. 62, pp. 653-661, Jun. 2026, [Online]. Available: https://doi.org/10.21595/vp.2026.26121.

[7] M. Shukurova, K. Ruziev, M. Talipov, and G. Talipova, "3D modeling of filtration in oil and gas reservoirs based on satellite data," Proceedings of International Conference on Applied Innovation in IT, vol. 14, no. 1, pp. 267-271, Mar. 2026, [Online]. Available: https://doi.org/10.25673/123569.

[8] M. Shukurova, M. Talipov, K. Ruziev, and K. Jurayeva, "Advanced geospatial monitoring of oil and gas infrastructure via satellite data," Mathematical Models in Engineering, vol. 12, no. 2, pp. 190-201, Jun. 2026, [Online]. Available: https://doi.org/10.21595/mme.2026.25328.

[9] Y. Croissant and G. Millo, "Panel data econometrics in R: The plm package," Journal of Statistical Software, vol. 27, no. 2, pp. 1-43, 2008, [Online]. Available: https://doi.org/10.18637/jss.v027.i02.

[10] S. Bird, E. Klein, and E. Loper, Natural Language Processing with Python. O'Reilly Media, 2009, [Online]. Available: https://doi.org/10.5555/1717171.

[11] M. Honnibal and I. Montani, "spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing," 2017, [Online]. Available: https://doi.org/10.5281/zenodo.1212303.

[12] K. Benoit et al., "quanteda: An R package for the quantitative analysis of textual data," Journal of Open Source Software, vol. 3, no. 30, p. 774, 2018, [Online]. Available: https://doi.org/10.21105/joss.00774.

[13] I. Feinerer, K. Hornik, and D. Meyer, "Text mining infrastructure in R," Journal of Statistical Software, vol. 25, no. 5, pp. 1-54, 2008, [Online]. Available: https://doi.org/10.18637/jss.v025.i05.

[14] Y. Xie, J. J. Allaire, and G. Grolemund, R Markdown: The Definitive Guide. Boca Raton, FL, USA: Chapman and Hall/CRC, 2018, [Online]. Available: https://bookdown.org/yihui/rmarkdown/.

[15] T. Kluyver et al., "Jupyter Notebooks – a publishing format for reproducible computational workflows," in F. Loizides and B. Schmidt, Eds., Positioning and Power in Academic Publishing: Players, Agents and Agendas. IOS Press, 2016, pp. 87-90, [Online]. Available: https://doi.org/10.3233/978-1-61499-649-1-87.

[16] P. Joyce, "Embedded Python," in C and Python Applications, pp. 151-181, 2021, [Online]. Available: https://doi.org/10.1007/978-1-4842-7774-4_5.

[17] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent Dirichlet allocation," Journal of Machine Learning Research, vol. 3, pp. 993-1022, 2003, [Online]. Available: https://doi.org/10.1162/jmlr.2003.3.4-5.993.

[18] B. Pang and L. Lee, "Seeing Stars: Exploiting Class Relationships for Sentiment Categorization with Respect to Rating Scales," in Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05), Ann Arbor, MI, pp. 115-124, 2005, [Online]. Available: https://doi.org/10.3115/1219840.1219855.

[19] Pyodide Developers, "Pyodide: Python distribution for the browser and Node.js based on WebAssembly," Version 314.0.2, 2026, [Online]. Available: https://pyodide.org/, [Accessed: Jul. 8, 2026].

[20] R-wasm Project, "webR - R in the Browser," Version 0.6.0, 2026, [Online]. Available: https://docs.r-wasm.org/, [Accessed: Jul. 8, 2026].