




cuallee: A python package for data quality checks across multiple DataFrame APIs

Herminio Vazquez ^{1*} and Virginie Grosboillot ^{2*}

¹ Independent Researcher, Mexico ² Swiss Federal Institute of Technology (ETH) * These authors contributed equally.

DOI: [10.xxxxxx/draft](https://doi.org/10.xxxxxx/draft)

Software

- [Review](#) 
- [Repository](#) 
- [Archive](#) 

Editor: 

Submitted: 10 March 2024

Published: unpublished

License

Authors of papers retain copyright and release the work under a Creative Commons Attribution 4.0 International License ([CC BY 4.0](https://creativecommons.org/licenses/by/4.0/)).

Summary

In today's world, where vast amounts of data are generated and collected daily, and where data heavily influence business, political, and societal decisions, it is crucial to evaluate the quality of the data used for analysis, decision-making, and reporting. This involves understanding how reliable and trustworthy the data are. To address this need, we have created cuallee, a Python package for assessing data quality. cuallee is designed to be dataframe-agnostic, offering an intuitive and user-friendly API for describing checks across the most popular dataframe implementations such as PySpark, Pandas, Snowpark, Polars, DuckDB and BigQuery. Currently, cuallee offers over 50 checks to help users evaluate the quality of their data.

Statement of need

For data engineers and data scientists, maintaining a consistent workflow involves operating in hybrid environments, where they develop locally before transitioning data pipelines and analyses to cloud-based environments. Whilst working in local environments typically allows them to fit data sets in memory, moving workloads to cloud environments involve operating with full scale data that requires a different computing framework ([Schelter et al., 2018](#)), i.e. distributed computing, parallelization, and horizontal scaling. cuallee accomodates the testing activities required by this shift in computing frameworks, in both local and remote environments, without the need to rewrite test scenarios or employ different testing approaches for assessing various quality dimensions of the data ([Fadlallah et al., 2023b](#)).

An additional argument is related to the rapid evolution of the data ecosystem ([Fadlallah et al., 2023a](#)). Organizations and data teams are constantly seeking ways to improve, whether through cost-effective solutions or by integrating new capabilities into their data operations. However, this pursuit presents new challenges when migrating workloads from one technology to another. As information technology and data strategies become more resilient against vendor lock-ins, they turn to technologies that enable seamless operation across platforms, avoiding the chaos of fully re-implementing data products. In essence, with cuallee no data testing strategy needs to be rewritten or reformulated due to platform changes.

One last argument in favor of using a quality tool such as cuallee is the need to integrate quality procedures into the early stages of data product development. Whether in industry or academia, there is often a tendency to prioritize functional aspects over quality, leading to less time being dedicated to quality activities. By providing a clear, easy-to-use, and adaptable programming interface for data quality, teams can incorporate quality into their development process, promoting a proactive approach of building quality in rather than relying solely on testing to ensure quality.

Methods

cuallee employs a heuristic-based approach to define quality rules for each dataset. This prevents the inadvertent duplication of quality predicates, thus reducing the likelihood of human error in defining rules with identical predicates. Several studies have been conducted on the efficiency of these rules, including auto-validation and auto-definition using profilers (Tu et al., 2023).

Checks

In cuallee, checks serve as the fundamental concept. These checks (Table 1) are implemented by rules, which specify *quality predicates*. These predicates, when aggregated, form the criteria used to evaluate the quality of a dataset. Efforts to establish a universal quality metric (Pleimling et al., 2022) typically involve using statistics and combining dimensions to derive a single reference value that encapsulates overall quality attributes.

Table 1: List and description of the currently available

Check	Description	Data Type
is_complete	Zero nulls	agnostic
is_unique	Zero duplicates	agnostic
is_primary_key	Zero duplicates	agnostic
are_complete	Zero nulls on group of columns	agnostic
are_unique	Composite primary key check	agnostic
is_composite_key	Zero duplicates on multiple columns	agnostic
is_greater_than	$col > x$	numeric
is_positive	$col > 0$	numeric
is_negative	$col < 0$	numeric
is_greater_or_equal_than	$col \geq x$	numeric
is_less_than	$col < x$	numeric
is_less_or_equal_than	$col \leq x$	numeric
is_equal_than	$col == x$	numeric
is_contained_in	$col \in [a, b, c, \dots]$	agnostic
is_in	Alias of is_contained_in	agnostic
not_contained_in	$col \notin [a, b, c, \dots]$	agnostic
not_in	Alias of not_contained_in	agnostic
is_between	$a \leq col \leq b$	numeric, date
has_pattern	Matching a pattern defined as a regex	string
is_legit	String not null & not empty $^{\wedge}\backslash S\$$	string
has_min	$\min(col) == x$	numeric
has_max	$\max(col) == x$	numeric
has_std	$\sigma(col) == x$	numeric
has_mean	$\mu(col) == x$	numeric
has_sum	$\Sigma(col) == x$	numeric
has_percentile	$\%(col) == x$	numeric
has_cardinality	$\text{count}(\text{distinct}(col)) == x$	agnostic
has_max_by	A utility predicate for $\max(col_a) == x$ for $\max(col_b)$	agnostic
has_min_by	A utility predicate for $\min(col_a) == x$ for $\min(col_b)$	agnostic
has_correlation	Finds correlation between 0..1 on $\text{corr}(col_a, col_b)$	numeric

Check	Description	DataType
has_entropy	Calculates the entropy of a column <code>entropy(col) == x</code> for classification problems	<i>numeric</i>
is_inside_iqr	Verifies column values reside inside limits of interquartile range $Q1 \leq col \leq Q3$ used on anomalies.	<i>numeric</i>
is_in_millions	<code>col >= 1e6</code>	<i>numeric</i>
is_in_billions	<code>col >= 1e9</code>	<i>numeric</i>
is_t_minus_1	For date fields confirms 1 day ago t-1	<i>date</i>
is_t_minus_2	For date fields confirms 2 days ago t-2	<i>date</i>
is_t_minus_3	For date fields confirms 3 days ago t-3	<i>date</i>
is_t_minus_n	For date fields confirms n days ago t-n	<i>date</i>
is_today	For date fields confirms day is current date t-0	<i>date</i>
is_yesterday	For date fields confirms 1 day ago t-1	<i>date</i>
is_on_weekday	For date fields confirms day is between Mon-Fri	<i>date</i>
is_on_weekend	For date fields confirms day is between Sat-Sun	<i>date</i>
is_on_monday	For date fields confirms day is Mon	<i>date</i>
is_on_tuesday	For date fields confirms day is Tue	<i>date</i>
is_on_wednesday	For date fields confirms day is Wed	<i>date</i>
is_on_thursday	For date fields confirms day is Thu	<i>date</i>
is_on_friday	For date fields confirms day is Fri	<i>date</i>
is_on_saturday	For date fields confirms day is Sat	<i>date</i>
is_on_sunday	For date fields confirms day is Sun	<i>date</i>
is_on_schedule	For date fields confirms time windows i.e. 9:00 - 17:00	<i>timestamp</i>
is_daily	Can verify daily continuity on date fields by default. [2,3,4,5,6] which represents Mon-Fri in PySpark. However new schedules can be used for custom date continuity	<i>date</i>
has_workflow	Adjacency matrix validation on 3-column graph, based on group, event, order columns.	<i>agnostic</i>
satisfies	An open SQL expression builder to construct custom checks	<i>agnostic</i>
validate	The ultimate transformation of a check with a dataframe input for validation	<i>agnostic</i>
iso.iso_4217	currency compliant ccy	<i>string</i>
iso.iso_3166	country compliant country	<i>string</i>
Control.completeness	Zero nulls all columns	<i>agnostic</i>
Control.percentage_fill	% rows not empty	<i>agnostic</i>
Control.percentage_empty	% rows empty	<i>agnostic</i>

References

- 52
- 53 Fadlallah, H., Kilany, R., Dhayne, H., El Haddad, R., Haque, R., Taher, Y., & Jaber, A.
54 (2023a). BIGQA: Declarative big data quality assessment. *Journal of Data and Information*
55 *Quality*, 15. <https://doi.org/10.1145/3603706>

- 56 Fadlallah, H., Kilany, R., Dhayne, H., El Haddad, R., Haque, R., Taher, Y., & Jaber, A.
57 (2023b). Context-aware big data quality assessment: A scoping review. *Journal of Data*
58 *and Information Quality*, 15. <https://doi.org/10.1145/3603707>
- 59 Pleimling, X., Shah, V., & Lourentzou, I. (2022). [Data] quality lies in the eyes of the beholder.
60 *Proceedings of the 15th International Conference on Pervasive Technologies Related to*
61 *Assistive Environments*, 118–124. <https://doi.org/10.1145/3529190.3529222>
- 62 Schelter, S., Lange, D., Schmidt, P., Celikel, M., Biessmann, F., & Grafberger, A. (2018).
63 Automating large-scale data quality verification. *Proc. VLDB Endow.*, 11(12), 1781–1794.
64 <https://doi.org/10.14778/3229863.3229867>
- 65 Tu, D., He, Y., Cui, W., Ge, S., Zhang, H., Han, S., Zhang, D., & Chaudhuri, S. (2023).
66 Auto-validate by-history: Auto-program data quality constraints to validate recurring data
67 pipelines. *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and*
68 *Data Mining*, 4991–5003. <https://doi.org/10.1145/3580305.3599776>

DRAFT